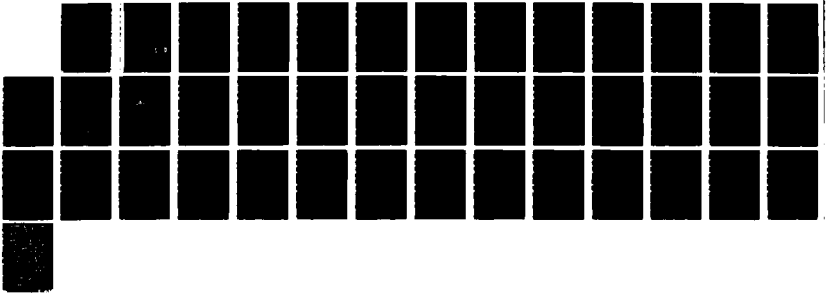
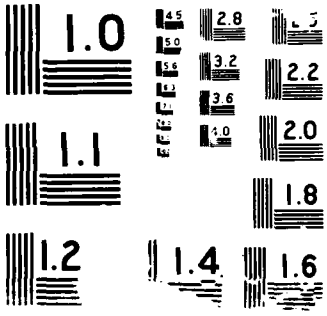
AD-A194 441 ESTIMATED PERFORMANCE OF A GATEWAY ROUTING-TABLE CACHE (U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE D C FELDMEIER MAR 88 MIT/LCS/TM-352 1/1

UNCLASSIFIED N00014-75-C-0661 F/G 25/5 NL




1.0
1.1
1.2
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6

LABORATORY FOR COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MIT/LCS/TM-352

ESTIMATED PERFORMANCE OF A GATEWAY ROUTING-TABLE CACHE

David C. Feldmeier

March 1988

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/TM-352	5 MONITORING ORGANIZATION REPORT NUMBER(S) N00014-75-C-0661 and N00014-83-K-0125
6a NAME OF PERFORMING ORGANIZATION MIT Laboratory for Computer Science 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Office of Naval Research/Department of Navy
6c ADDRESS (City, State, and ZIP Code) 545 Technology Square Cambridge, MA 02139	7b ADDRESS (City, State, and ZIP Code) Information Systems Program Arlington, VA 22217
8a NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22217	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification)
ESTIMATING PERFORMANCE OF A GATEWAY ROUTING-TABLE CACHE

12 PERSONAL AUTHOR(S)
Feldmeier, David C.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM ____ TO ____	1988 March	37

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	computer network interconnection, gateways, caches, performance measurement, cache simulation

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Gateway throughput can be increased by reducing the routing-table lookup time per packet. A routing-table cache can be used to reduce the average lookup time per packet and the purpose of this paper is to determine the performance of such a cache. The performance results of cache simulations driven by measured traffic data for gateways at MIT are presented. These results include the probability of reference versus previous access time, cache hit ratios, and the number of packets between cache misses. A simple, conservative analysis using the presented measurements shows that current gateway routing-table lookup time could be reduced by up to 77%.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Judy Little, Publications Coordinator	22b TELEPHONE (Include Area Code) (617) 253-5894 / 22c OFFICE SYMBOL

Estimated Performance of a Gateway Routing-Table Cache

David Charles Feldmeier

March 1988

This research was supported in part by the Defense Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract numbers N00014-75-C-0661 and N00014-83-K-0125.

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

Estimated Performance of a Gateway Routing-Table Cache

David C. Feldmeier
Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Ma. 02139

Abstract

Gateway throughput can be increased by reducing the routing-table lookup time per packet. A routing-table cache can be used to reduce the average lookup time per packet and the purpose of this paper is to determine the performance of such a cache. The performance results of cache simulations driven by measured traffic data for gateways at MIT are presented. These results include the probability of reference versus previous access time, cache hit ratios, and the number of packets between cache misses. A simple, conservative analysis using the presented measurements shows that current gateway routing-table lookup time could be reduced by up to 77%.

Keywords: computer network interconnection, gateways, caches, performance measurement, cache simulation.

1. Introduction

A gateway is a device that connects two or more heterogeneous networks and is responsible for transporting packets among them, assuring that packets arriving from one network are retransmitted toward their ultimate destination. Any gateway that is not directly connected to the destination must send the packet to the next gateway in the chain from source to destination; the address of this next gateway is the next-hop address. The gateway examines the destination address of a received packet and determines the next hop address by using the destination address as an index to a *routing table*. A routing table is a table of destination/next-hop address pairs that is maintained by the gateway. A routing-table lookup converts an internet destination address into a local network address. If the packet destination is on the same network as the gateway, then the next-hop is the local network address of the destination. Otherwise, the next-hop is the local network address of the next gateway that brings the packet closer to its final destination. The gateways examined in this study forward TCP/IP packets.

High-throughput gateways are necessary in an inter-network environment to provide high-speed communication that is transparent to the user. In the near future, we will have networks operating at 100 Mbps or more. A fully utilized gateway connecting two 100 Mbps networks transporting 500 byte packets would have only 40 μs to handle each packet. An essential process of packet forwarding is using the gateway's routing table to determine the next-hop destination of a packet. The routing table lookup time must be reduced to achieve high throughput gateways.

In computer systems, average memory access time is decreased economically with a cache. For a cache to be effective, the overhead of checking the cache must be less than the savings in memory access time due to cache hits. This implies that a locality of references to memory locations must exist for the

cache to be effective. Two types of reference locality are known for computer memory applications: temporal locality and spatial locality. Temporal locality of reference means that the probability of reference to a memory location is related to the time of previous references. Spatial locality means that the probability of reference to a memory location is related to the references to memory locations near the one in question. If locality exists for packet destination addresses, then routing-table caching of destination/next-hop pairs will decrease the average processing time per packet and increase gateway throughput.

Previous measurements on a token ring local area network at the MIT Laboratory for Computer Science (LCS) show that packet arrivals on the network are not random; the probability of a packet arriving shortly after a given packet is higher than that given by a random (Poisson) packet generating process, which implies that packets arrive in groups [3]. Additional measurements by Jain and Routhier showed that if a packet is seen from A to B, there is a 29% chance that the next packet is from A to B and a 31% chance that the next packet is from B to A [5]. The clustering of packets is caused by file transfers and virtual circuit connections. For a file transfer, a file is divided onto multiple packets which are transmitted as quickly as possible, forming a burst of packets. Consequently, if a packet is seen from host A to host B, then the next packet has a high probability of being from A to B. During a virtual circuit connection, packets are acknowledged by the receiver and this acknowledgment is transmitted shortly after a packet is received, forming groups of packet-acknowledgment pairs. Thus, a packet from A to B has a high probability of being followed by a packet from B to A. The packet address locality observed is for the aggregate of all hosts on the network, and packet address locality for any particular host must be at least this good. Although the measurements by Jain are of local network addresses, if file transfers and virtual circuit connections are responsible for address locality, then packet address locality should exist also at the internet level because these services are built using an internet protocol.

In summary, the routing-table lookup is a time-consuming process of packet forwarding. Routing-table caching of destination/next-hop address pairs will decrease the average processing time per packet if locality exists for packet addresses. Therefore, gateway throughput can be increased if internet address locality exists and earlier measurements suggest that this is likely. The purpose of this research is to determine whether internet address caching in gateways is effective and the expected cache performance. This research simulates the performance of several routing-table caches in busy gateways, using recorded gateway traffic to drive the simulation. The following sections include a discussion of the cache simulation models, the collection of data that drive the cache simulation, the simulation results and analysis, a cache performance analysis, and conclusions. This paper is a revised, extended version of an earlier paper by Feldmeier [4].

2. Data Generation

The method used for generating data to determine the effectiveness of routing-table caching is *trace-driven simulation*. A trace is gathered for an operating gateway by recording every routing table reference during normal gateway operation. This trace is then used to drive a cache simulation model. The cache model can be altered to simulate a cache of any type and size, and the simulation can be repeated on the same data set.

2.1. Cache Modeling

2.1.1. Cache Location

Determining the next-hop address of a packet based on its destination address is more complex than a simple routing-table access on current gateways. Currently, routing is based on a hierarchical addressing scheme, which means that some part of the destination address contains information about what network the destination host is on. The routing table contains information only about the next hop to reach a given network, rather than a given destination. Before the routing table is used, the network address for the destination must be extracted from the destination address. The decoding of the destination address into a network address and a host address is non-trivial because the network address may be encoded in the destination address in different ways.

Two possible positions for a cache are before the address decoder or between the address decoder and the routing table. The advantage of placing a cache before the address decoder is that destination addresses found in the cache need never be processed by the decoder. The disadvantage is that for a hierarchical addressing scheme, the number of addresses exceeds the number of networks, so for a cache of fixed size, the cache performance for the full address must be no better than the performance of a cache for the network part of the address. The choice of cache location is determined by the relative cache performance for flat and hierarchical addresses and by the time required for address decoding. In fact, some simple pre-processing of the destination address in such a way that does not affect address decoding may increase the performance of the cache placed before the address decoder at relatively little cost. The cache/pre-processor combination before the address decoder may offer the best overall performance.

2.1.2. Cache Simulation

A routing-table cache differs slightly from a main memory cache. A memory cache will pull data into the cache on either a read or a write. Since the only reason to write to the routing table is because new routing information has arrived, routing table writes have no address locality. Therefore a routing table cache should only pull entries into the cache on a read, never on a write. This means that updating the routing table is now slightly more expensive, as each write requires accessing the cache, but it also means that any currently active entries in the cache will remain valid and thus the cache does not have to be flushed. The cache model assumes that the cost of updating the routing table is negligible, which approximates true cache operation if the routing table is quasi-static. Current gateway routing tables are relatively static, so the assumption of negligible routing table update cost is justified.

Different types of caches will behave differently with the same data set, so a type of cache to simulate must be chosen. A routing-table cache is defined by its associativity, replacement, and prefetch strategies, as well as size. Each of these must be defined for the simulation model.

The associativity of a cache ranges from fully associative to direct mapped. Fully associative caches allow any destination/next-hop pair to be stored in any cache location. A direct-mapped cache allows each destination/next-hop pair to be stored in only a certain cache location, so multiple destination addresses are mapped into a single cache location. In general, the cache is divided into a disjoint set of locations that may contain the contents of a certain subset of destination/next-hop pairs, and this is a set-associative cache. For these measurements, a fully associative cache is used because the performance of any set-associative cache depends on the value of those items to be stored in the cache and the cache association mechanism. A fully-associative cache avoids the problem of measurement bias because of specific network addresses in the measurements and makes the results general for any set of network addresses with similar characteristics. In addition, broad associative searches are easily implemented in

MOS VLSI, so it is reasonable to expect that highly associative caches will be available on a chip [6].

Whenever there is a cache miss, the missed entry must be entered into the cache and some other item must be discarded. The three most popular replacement strategies are random replacement, first-in first-out (FIFO), and least recent use (LRU). A random replacement strategy simply discards a random location in the cache. The "random" location is most easily chosen based on some counter value (such as a clock) from elsewhere in the system. A FIFO replacement strategy discards the cache entry that is the oldest; a FIFO cache is easily implemented as a circular buffer. The most complex strategy, but also the one that generally provides the best cache performance, is LRU. An LRU cache operates by storing not only the data, but also the time that each datum was last referenced. If a reference is in the cache, its time is updated; otherwise the data that was least recently accessed is discarded and the data for the new reference is cached. For most of the measurements in the paper, LRU and FIFO caches are modeled. Full associativity maximizes the performance of LRU caches; this is not necessarily true for FIFO caches.

An LRU cache is a good choice for the type of address locality expected, but unfortunately this type of cache is also the most complex and expensive to implement. A different type of cache may perform worse for a given number of slots, but for a fixed price, a less efficient cache can afford to have more slots and perhaps exceed the performance of a smaller LRU cache; this is a cost trade-off that only the gateway designer can assess.

Another possibility is to use a combination of two or more caches, for example a LRU cache followed by a FIFO cache. If an entry were found in neither cache, then the new address would be put into the LRU cache, the element discarded from the LRU cache put into the FIFO cache and the FIFO discard removed. If the entry were found in the LRU cache, the LRU cache would be reordered and the FIFO cache left alone. If the entry were found in the FIFO cache, the entry should be added to the LRU cache and the discard from the LRU cache should replace the original entry in the FIFO cache, so that any entry is represented at most once in the two caches.

A strategy for prefetching is suggested by observing that a packet from host A to host B is often followed by a packet from B to A [5]. If the cache prefetches the next-hop for the packet source, then a packet traveling through the same gateway in the reverse direction no longer causes a cache miss. One way to prefetch is to do a routing-table lookup on the packet source address. Since the probability of a reverse-direction packet is less than one, waiting until a packet arrives in the reverse direction before doing a routing-table lookup will always give better performance. Another way to prefetch is to use the local network source of an incoming packet as the next-hop address for packets going in the opposite direction. However, this only works if routing is completely bidirectional (at least through the gateway in question). Although bidirectional behavior is seen on current gateways, there is no reason why this must continue to be true. Even if a prefetch is inexpensive, there is the possibility of forcing an active entry out of the cache. This implies that the probability of a reverse-direction packet is large or that the cache must hold up to twice as many entries as the number of simultaneously active trains for prefetching to be effective.

For the cache simulations, two situations are analyzed. The first is that only the packet destination is used to update the cache, corresponding to a case where prefetching is not economical. The second assumes that prefetching is free and should be done any time that the packet source is not in the cache. In reality, the truth is somewhere in-between, so these two sets of curves give lower and upper bounds for fully associative LRU caches where prefetching occurs.

2.2. Trace Generation

We would like to estimate the performance of a gateway with a routing-table cache before the system is built. The performance can be determined by trace-driven simulation, which requires a list of all routing-table accesses on a gateway. We are interested in statistics of packet arrivals at various gateways, but building a measurement system directly into a gateway has several disadvantages. The main problem is that each gateway would need to be reprogrammed to include a monitoring system. Reprogramming is difficult because some of the gateways are owned by other research groups, which are reluctant to disturb gateway operation; other gateways are commercial products, in which case the source code is unavailable. In addition, any monitoring system installed in a gateway will use processor time and memory space and this overhead could affect gateway operation during periods of heavy load, causing packets to be dropped. Since periods of heavy load are of particular interest for determining the efficiency of the caching system, the measurement artifact of such a monitoring program is unacceptable. Also, it is advantageous to be able to try various caching strategies and cache sizes on a single measurement set, so that differences in results are directly attributable to the change in strategy, rather than simply a change in the gateway traffic. Instead of monitoring the packet arrivals at each gateway, it is simpler to measure all of the packets on the network that pass through the gateway. This replaces monitoring programs in each gateway with a single dedicated monitoring system.

Trace-driven simulation computes the exact cache performance as long as the trace-gathering operation is exact and has no processing cost. Measuring packets on the network with a separate measurement system assures that the gateway itself operates as usual. Two types of packets are observed traveling to a gateway: packets to be forwarded to another network or packets destined to the gateway itself. Not all packets addressed to the gateway itself are observed, since only one of the networks attached to the gateway is monitored, but this is a problem only if a packet to a gateway causes a routing table reference. Whether a packet to a gateway causes a routing table reference depends on the gateway implementation. One possibility is to explicitly check each incoming packet against a table of all of the gateway's addresses to see if there is a match. This explicit check means that the routing table is never consulted about packets destined to the gateway. Another possibility is to use the routing table for all packets. The gateway will discover that it is directly connected to the proper network and will try to send the packet. Before the packet is sent, the gateway checks if the packet is to its own address on the appropriate network. If the packet is for the gateway, then it is never transmitted. The explicit check after the routing-table lookup requires checking a smaller number of gateway addresses at the increased cost of a routing-table lookup. For this paper, it is assumed that gateways do an explicit address check and that the unobserved packets do not cause routing-table accesses.

A possible problem with a routing-table trace generated from packet measurements on the network is that packets are seen at the network measurement system at different times than they are looked up in the routing table. This time difference is caused by propagation delays and queueing times between the gateway and the network measurement system. Any delay greater than zero allows a pair of packets in the trace to be reordered if they arrive at the gateway almost simultaneously from opposite directions. Increasing the delay allows packets to be displaced by more than one position in the trace. Packet displacement in the trace can affect the cache hit ratio of a cache; reordered elements near the top of the cache will not change the cache performance, but reordered elements at the bottom of the cache may be expelled prematurely. As long as adjacent addresses in the cache have an approximately equal chance of being used (as they would near the bottom of an LRU cache), then the cache hit ratio should not be affected much.

A packet arriving at the gateway is placed in an input queue. Packets are removed from the input queue

by the packet-forwarding process; after a routing decision is made, the packet is then placed in the appropriate output queue. The network measurement system sees packets to the measured network after the routing table, the time difference equal to the waiting time in the gateway output queue for the ring network plus the propagation delay between the gateway and the network measurement system. The measurement system sees packets from the measured network before the routing table, the time difference equal to the propagation delay between the gateway and the measurement system and the waiting time in the gateway input queue for the ring network. The ring network is lightly utilized and thus any packets in the gateway output queue should be transmitted almost immediately. The input queue from the ring should also cause minimal delay because any packets from the ring are handled immediately and placed in the output queue for the other network attached to the gateway. As long as this delay is small, packets in the trace will seldom be more than one place out of order.

Another disadvantage of measuring all packets to a gateway is that the packets on the network to (or from) a gateway may not be the same as those to be handled by the gateway software. If the current gateway drops a packet, some protocols will retransmit that packet. Thus our packet trace now has multiple copies of a single packet, only one of which is actually forwarded by the gateway. Some of these extraneous packets can be eliminated by filtering the packet trace to remove identical packets with certain interarrival times. In any case, these duplicate packets should only appear when the gateway is overloaded, and this is relatively rare.

A limitation on the gateways that may be observed with network measurements is also caused by unobserved packets. If a gateway completely connects N networks, then there are $\binom{N}{2}$ distinct paths through the gateway. By monitoring a single network, the packets entering and exiting a single interface are observed and these are packets to or from $N-1$ other networks. Therefore, of the $\binom{N}{2}$ paths through the gateway, the monitoring system can only monitor $N-1$ of these paths. Notice that for $N=2$, all paths are observable; this means that the complete set of routing-table accesses for an existing gateway is measurable by network monitoring of a single network if the gateway connects exactly two networks. The gateways chosen for most of the measurements presented in this paper connect exactly two networks.

Measurements of gateways that connect three or more networks still provide useful information. Since these measurements are not for improving a specific gateway but rather for determining whether gateways in general might be improved, the idea of a virtual gateway is useful. Any gateway that connects N networks can be considered to be N dual-interface virtual gateways, each of which connects one of the networks to a central, shared virtual network. The measurements provide information about the one virtual gateway that connects the measured network to the virtual network. This virtual network in turn connects to the other $N-1$ networks via $N-1$ virtual gateways.

The measurement system on the token ring is a general measurement system for monitoring all traffic and was originally built for the research done by Feldmeier [3]. The monitoring system consists of two computers - a passive monitor and a data analysis machine. The passive monitor receives the packets on the ring and timestamps them upon arrival. The passive monitor then compresses the information of interest from each packet into a large data packet that is sent to the analysis machine. The monitoring system generates less than 2% of the network traffic, so the monitoring overhead is acceptable; also, none of the monitoring system packets pass through a gateway, so the arrival order of packets at a gateway remains unchanged. A more complete description of the design of the network monitoring station may be found in a paper by Feldmeier [2]. For these measurements, the monitor compresses information only for packets that are to or from non-local hosts; in other words, all the packets that pass through the gateways.

The analysis machine simply stores all of the collected data for later analysis.

Only a single trace using complete [illegible] for all packets through the gateway is needed for both flat and hierarchical addressed caches for the simulation. Flat addressing measurements use the entire internet address for the cache simulation, hierarchical addressing uses only the network field of the internet addresses for the cache simulation.

2.3. Data Analysis

The measurements used in this paper were collected on a 10 Mbps token ring at the MIT LCS. In addition to 8 gateways, 37 computers also reside on the ring, including 7 VAXs, 7 PCs, a microVAX, an IBM-[illegible] and [illegible] The measurements are based on network measurements made during the 24 hour period from 00:00 Wednesday, February 3rd to 00:00 Thursday, February 4th 1988. It is for this data that cache performance is evaluated, however any cache must start with no data and each cache reference that initializes the cache causes a cache miss. Since the reason for installing a cache in a gateway is to improve long-term average performance of the system, we are interested in the steady-state cache performance. To avoid cache misses caused by cache initialization, the cache is pre-loaded so that once the measurement period begins, the cache has many of its most referenced entries. Measurements during the 24 hour period from 00:00 Tuesday, February 2nd 1988 to 00:00 Wednesday, February 3rd 1988 were used to pre-load the cache to avoid measurements during cache initialization. The total number of packets carried by the ring during the 24 measurement period was 6,140,576 packets.

A busy gateway is the ARPANET gateway (local network address 4). This is the main ARPANET connection for MIT and it serves the entire campus as well as several local companies involved in research. The two network interfaces of the gateway connect to the ring and the ARPANET. The ARPANET gateway processed 2,847,682 packets during the 24 hour measurement period. During cache initialization, 1426 distinct destination addresses and 1547 distinct destination and source addresses arrived, out of 2,374,412 packets. During the day of measurement, an additional 415 destination and 472 destination/source addresses were observed.

A gateway connects the ring to the main LCS Ethernet (local network address 6). Many of the computers in LCS are connected either directly or through repeaters to the LCS Ethernet. The LCS Ethernet gateway processed 1,062,294 packets during 24 hours. During cache initialization, 636 distinct destination addresses and 662 distinct destination and source addresses arrived, out of 1,058,834 packets. During the day of measurement, an additional 176 destination and 187 destination/source addresses were observed.

The busiest gateway on the ring is the gateway that connects to the main campus ring (local network address 7). The main campus ring is a fiber-optic token ring that circles MIT and acts as a spine network for the many local area networks across campus serving a few thousand machines. The three network interfaces of the gateway connect to the ring, the main campus ring, and the main LCS Ethernet. The main campus gateway processed 3,769,544 packets in 24 hours.

The MIT Artificial Intelligence Laboratory has its own extensive computer network system and this is connected to the LCS token ring as a local network address 9). The four network interfaces of the gateway connect to the ring, and three AI laboratory Ethernets. The AI gateway processed 914,570 packets during 24 hours.

A fifth gateway that carries much traffic connects the ring to a 10 Megabit Ethernet of the Multiprocessor Emulation Facility (local network address 11). The three network interfaces of the gateway connect to the ring, and two Ethernets. This gateway processed 1,067,433 packets during 24 hours.

As with any measurement, the data collected is not perfect. The monitor transmits information about each packet to the analyzer once to minimize network loading, and if there is a network error, the monitoring data are lost. During the course of monitoring, data were lost for 0.027% of packets on the ring.

As mentioned above, the monitor cannot receive packets addressed to the gateway or broadcast packets from other networks, and it is assumed that these packets do not cause routing table accesses. Any packets of these types received on the monitored network are ignored in the data analysis.

Packets in the trace may be reordered because of the delay between the gateway and the network measurement system. Any packets that are out-of-order at the network measurement system must include a packet to the gateway closely followed by a packet from the gateway, because one packet must be seen first by the measuring system and the other must be seen first by the gateway. For packets arriving simultaneously at the ARPANET gateway, a pessimistic estimate of the time before the packet from the ARPANET is seen by the network measurement system is 4 ms. During the 48 hour preload and measurement period, the percentage of packets through the ARPANET gateway that are within 4 ms of each other as described above is 5.8%. Thus up to 5.8% of all address pairs in the trace could possibly be reordered. The overall effect of these reorderings is probably negligible, since other effects external to the gateway (such as queueing and packet loss) would cause packet reordering during normal gateway operation.

even if every reordering cause a hit (miss), the cache hit ratio would not increase (decrease) by more than 1.76%. Since millions of packets are in each routing-table trace, the effects of reordering will probably be negligible over the long run.

Another source of measurement inaccuracies is caused by low-level retransmissions. As mentioned above, not all packets on the network are seen by the intended receiver. To assure more reliable packet transmission, the ring has a low-level acknowledgment system built into the network hardware; if an interface receives a packet, an acknowledgment bit in the packet trailer is marked. Since the transmitter must remove any packets that it transmits on the ring, the transmitter always can determine whether the packet it has sent was received by checking for the receiver's mark. The device drivers for hosts on the ring include a low-level retransmit mechanism that automatically retransmits a packet up to eight times without higher-level intervention if the receiver has not acknowledged the packet by marking it. This means that several closely-spaced identical packets may be transmitted, but only a single packet is received by the intended host. The monitoring station is fast enough to receive all of the packets present in a burst, but the receiver itself receives only one of them. The monitoring station cannot check the acknowledgment bit, so somehow these retransmissions must be removed from the data so that the time locality of the data will not be exaggerated. Since a low-level packet burst is at high speed, any packets with the same source and destination address that arrive within a short time of each other should be discarded; however, if this threshold is set too high, multiple packets between a host-destination pair may be incorrectly eliminated. The specific value that determines whether a packet is caused by a low-level retransmission depends on parameters of the network and the network hosts. Also, the threshold should exceed the transmission time of most packets. The longest commonly used packets on the ring are 1082

bytes long, and at 10 MBPS the transmission time for these packets is 868 microseconds.

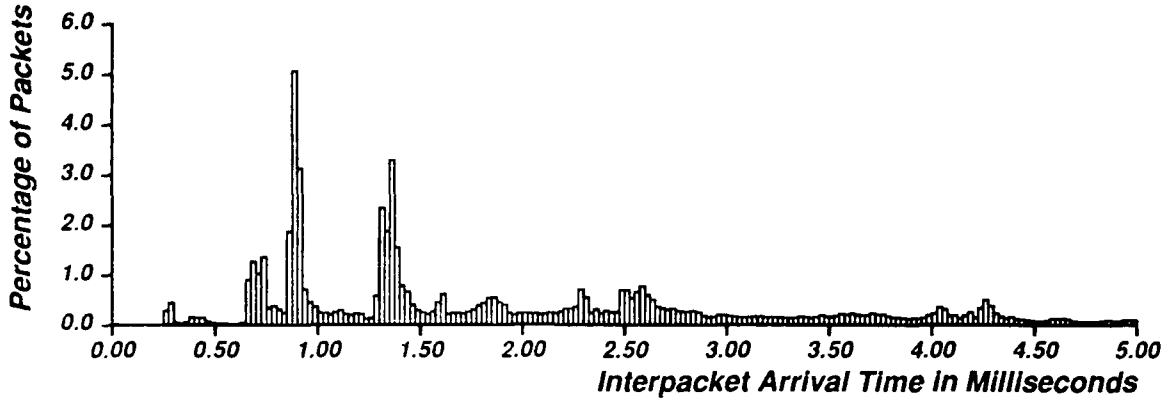


Figure 2-1: Interpacket Arrival Time of Identically Addressed Packets

The graph in figure 2-1 is a histogram of the interpacket arrival times for packets to all gateways with identical source and destination addresses, from 0 to 5 milliseconds with a clock granularity of 25 microseconds. There are peaks around 700 and 900 microseconds, and this is where most of the low-level retransmission must occur. The higher the low-level retransmission detection threshold is set, the more low-level retransmissions that are eliminated. However, the upper bound of this threshold is set by the interpacket time for the fastest host on the network. Since a gateway can transmit over 700 packets per second, the interpacket time of 1.35 milliseconds (the peak around 3% of all packets) is probably caused by gateway transmissions. For this data, any packets with the same source and destination that arrive within a millisecond or less of each other are discarded as low-level retransmissions.

3. Measurement Results

The behavior of the cache is more easily understood for flat addressing than hierarchical addressing because hierarchical addressing maps individual flows into a conglomerate of network flows. Although hierarchical addressing is relevant to existing gateways, there are a number of reasons why flat address caching should be examined. For hierarchical addresses, a flat address cache hit saves not only a routing-table lookup but also a flat-to-hierarchical address mapping. Another advantage of retaining the flat address is that the gateway can manipulate data based on individual packet flows from a source to a destination, for functions such as billing, access control and congestion control.

3.1. Results for Flat Addresses

The cache performance for flat addresses depends on the relative popularity of various addresses on packets that pass through the gateway. As can be seen from the graphs for the ARPANET gateway (gateway 4) and the LCS Ethernet gateway (gateway 6) in figures 3-1 and 3-2, relatively few addresses account for the majority of packets that pass through a gateway - a sure sign that a cache could be effective.

Figure 3-3 shows the distribution of the number of simultaneously active packet trains computed after each packet arrival for the ARPANET gateway. An active train is defined as a train whose interpacket arrival times do not exceed 10 seconds. If no packet is received from a train for 10 seconds, a train is considered inactive and the time of deactivation is the time of the last packet arrival for that train. The same graph for the LCS Ethernet gateway is seen in figure 3-4.

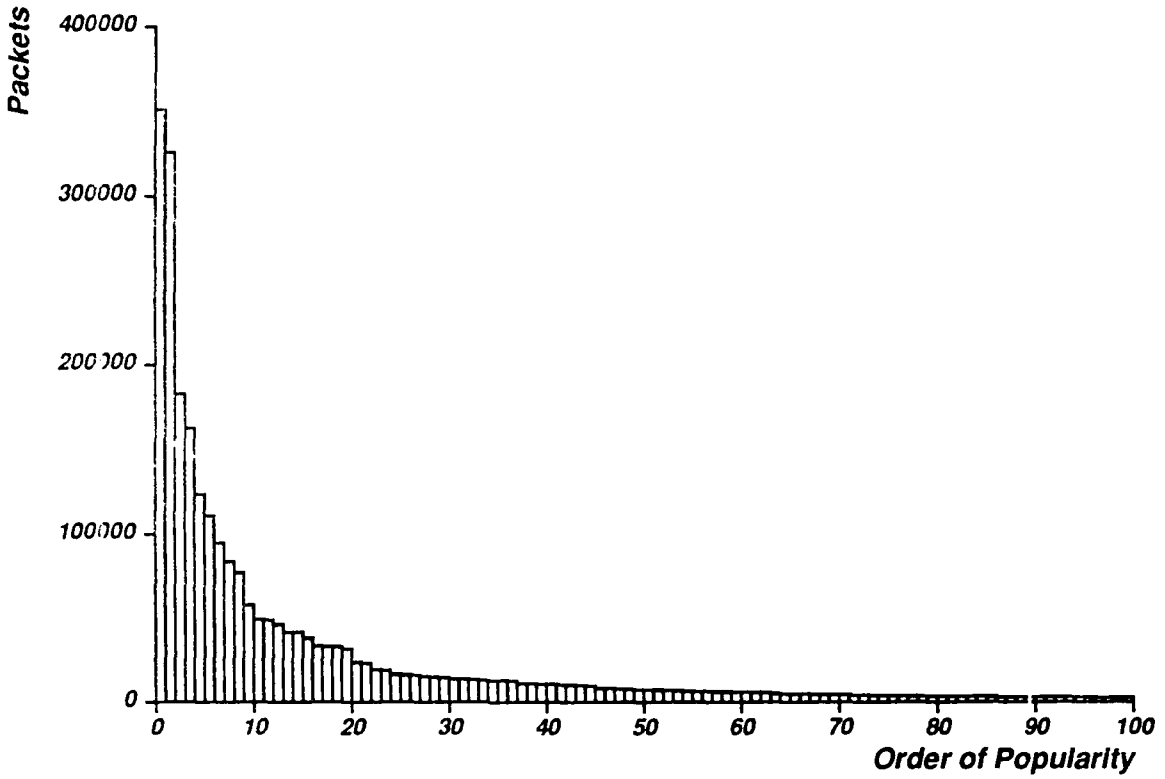


Figure 3-1: Number of Packets Ordered by Popularity (Gateway 4)

From a previous study, it is known that a packet from host A to host B has a high probability of being followed by another packet from A to B [5], which implies time locality of packet addresses. Consider a time-ordered list of previously seen addresses; for a current address that has been previously seen, how far down the list is this address? Figure 3-5 shows the percentage of references versus time of previous reference for the 400 most recently referenced packet destination addresses for the ARPANET gateway. Figure 3-6 is identical to figure 3-5 except that both packet destination and source addresses are used. The curves are plotted on a logarithmic scale so that the relative popularity of the slots is more easily seen.

The decrease in probability for both graphs is nearly monotonic until about 100 most recently referenced addresses. This monotonically decreasing function implies that the LRU cache management strategy will produce the best hit ratio for caches of less than 100 slots. After reference 100, the probability continues to decrease generally, but there is more variance. One reason for the increased variance is that the probability of reference decreases with increasing age, so that the amount of data that falls into the farther positions is too small to properly approximate the probability distribution curve at these points. At these outer parts of the curve, the average number of packets per slot is about 15, as compared to between 180 and 750,000 in the first 100 positions. Another explanation is that the relationship between probability of reference and time of last reference changes at some point on the curve. This change in relationship might be caused by two separate mechanisms that determine whether the current packet address is in the cache.

The first mechanism that produces cache hits is a packet that is part of an active train of packets. The first packet of the train initializes the cache and as long as the cache is not too small, all following packets are cache hits. The second mechanism is that the first packet of a train is to a destination already in the

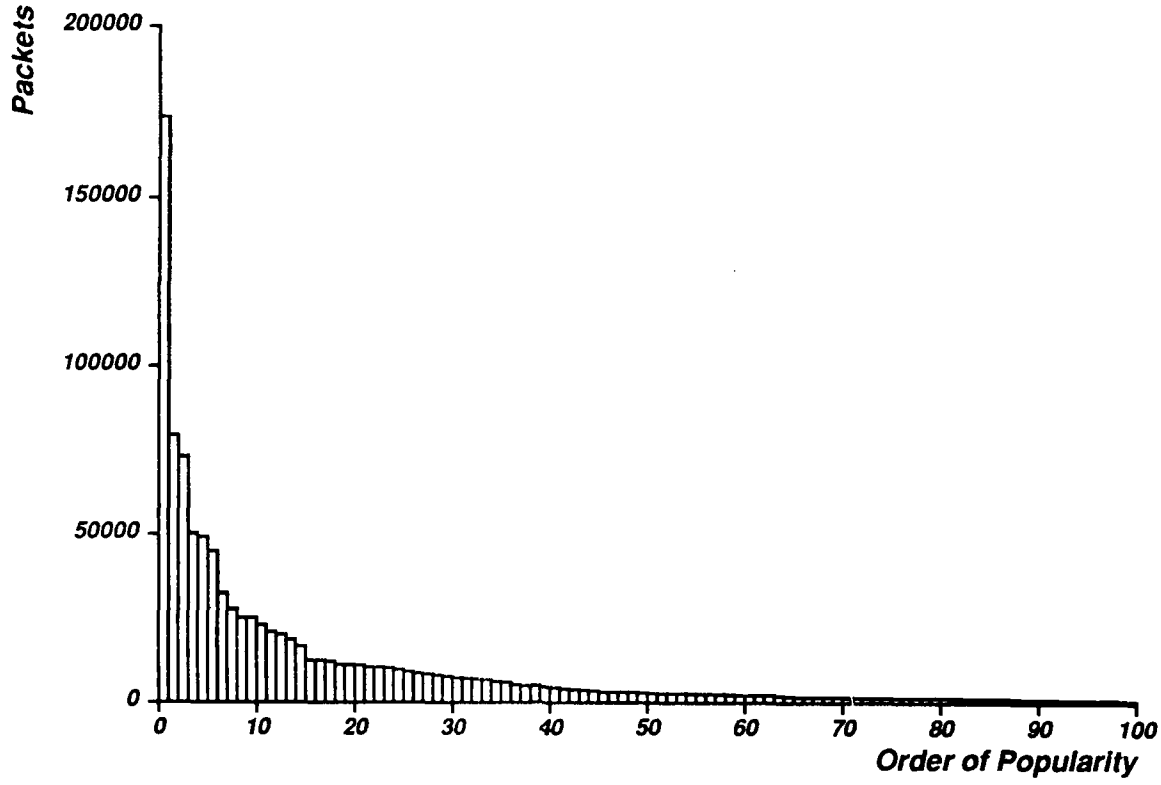


Figure 3-2: Number of Packets Ordered by Popularity (Gateway 6)

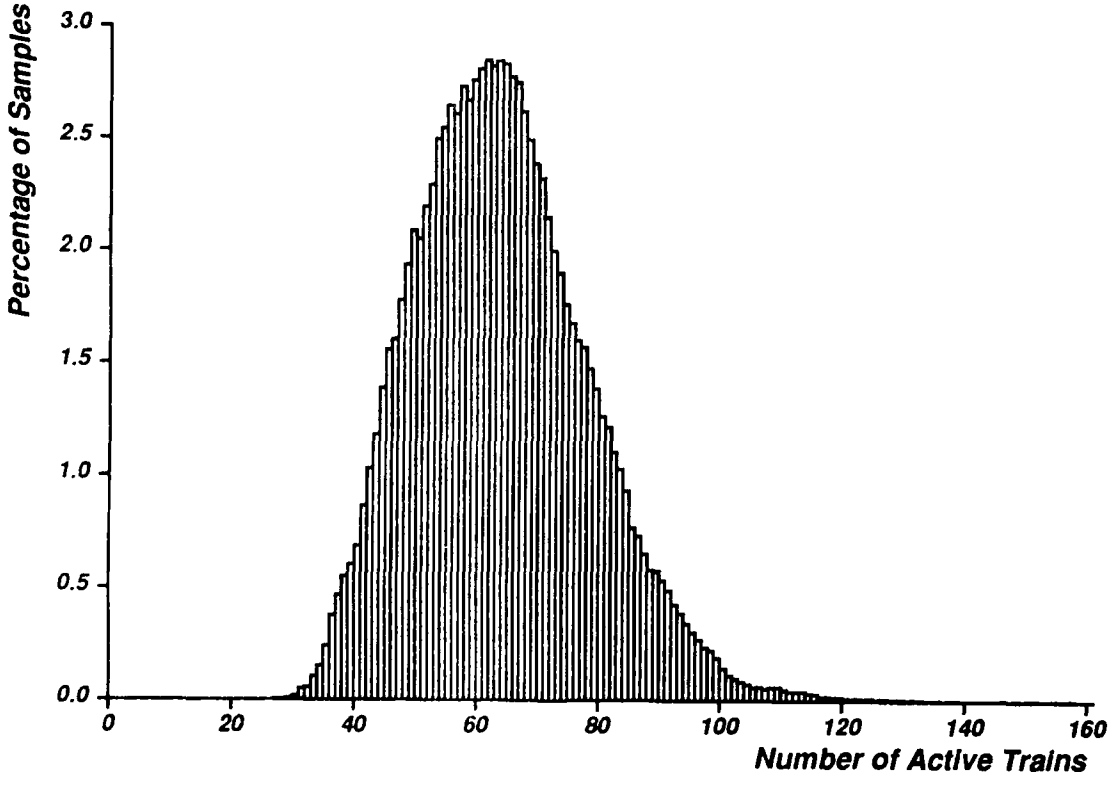


Figure 3-3: Percentage of Simultaneously Active Trains (Gateway 4)

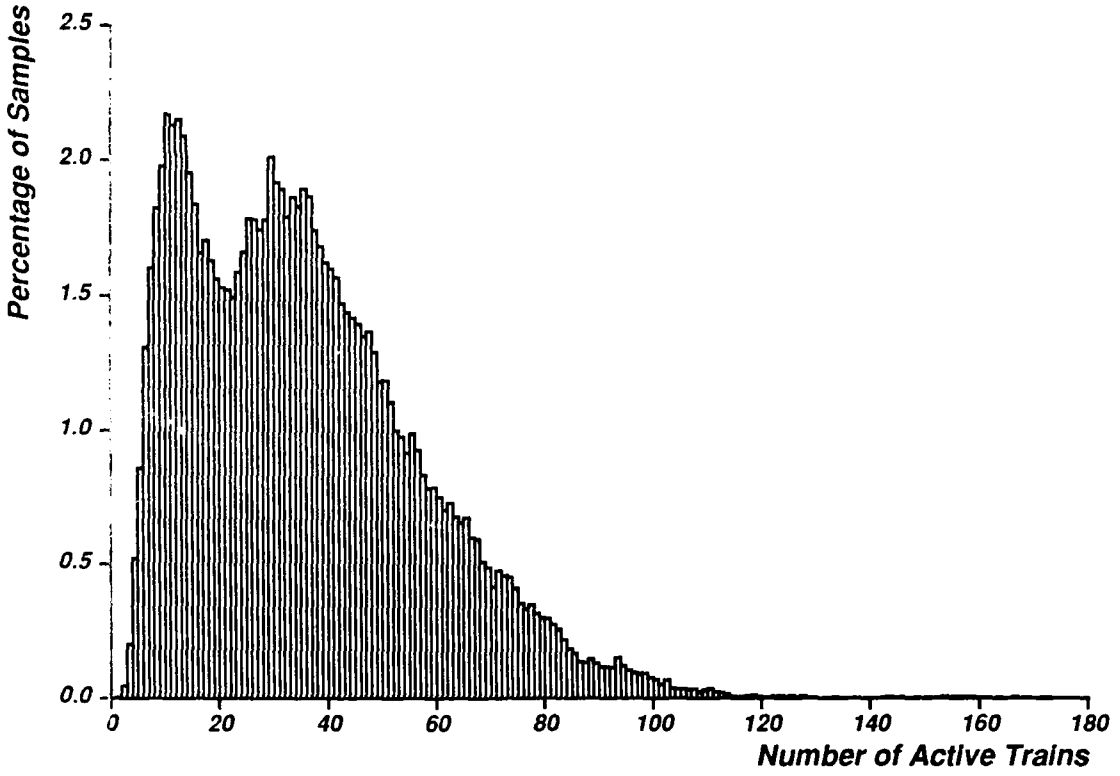


Figure 3-4: Percentage of Simultaneously Active Trains (Gateway 6)

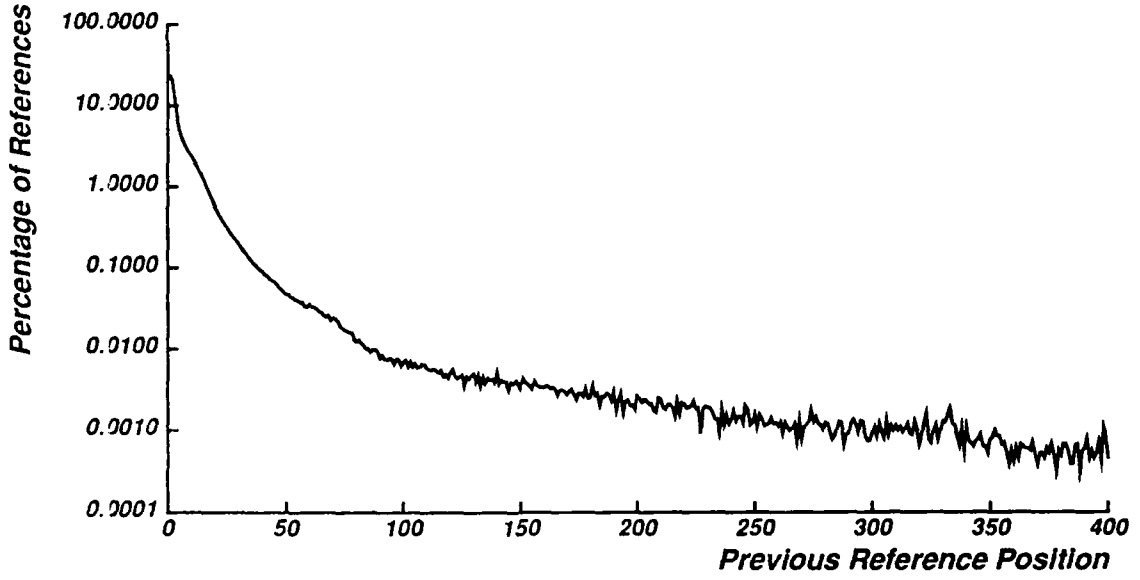


Figure 3-5: Percentage of Reference versus Time of Last Reference (Destination - Gateway 4)

cache. If N is the average number of packets in a train[1], the number of packets that arrive due to the first mechanism must be greater than N-1 times the number of packets that arrive due to the second

[1] Jain and Routhier claim the average number of packets in a train is 17.4 [5].

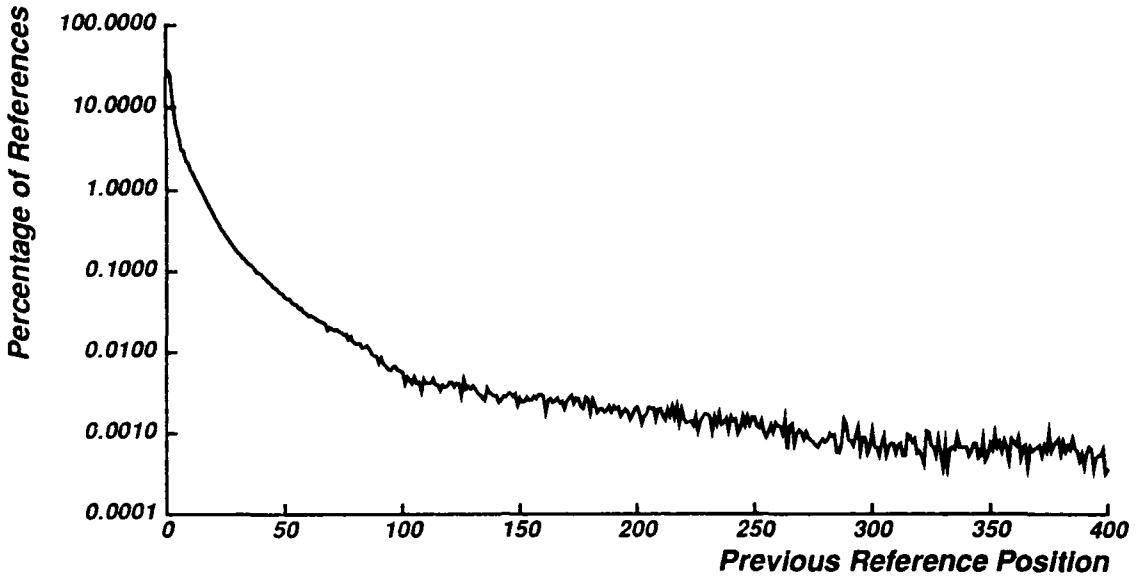


Figure 3-6: Percentage of Reference versus Time of Last Reference (Destination/Source - Gateway 4)

mechanism[2]. This means that the destination addresses of packets in an active train are most likely to be near the first slot of an LRU cache and that addresses of inactive but popular train destinations are likely to be between slot *i*, where *i* is the number of simultaneously active trains, and the last slot of the cache. The relationship among train arrivals at a destination is unknown, but undoubtedly it is weaker than the relationship among packets in a train, and this could explain the higher variance in the later part of the probability distribution. Additional information is necessary to decide whether an insufficient number of samples or two cache hit mechanisms more correctly explains the increased variance in the later part of the probability distribution above.

If the probability distribution does change with age after a certain point, the most cost-effective strategy may be to have an LRU cache with 100 slots, followed by a FIFO or random management cache with a few hundred slots. If the probability of reference does not decrease monotonically with increasing last-reference age, then an LRU cache decreases in efficiency relative to other cache types and its higher cost may not be justified because other cache types allow a larger cache for the same price and perhaps a higher cache hit ratio for a given cost.

The two graphs for the LCS Ethernet gateway are shown in figures 3-7 and 3-8. The waviness of the lines are indicative of a periodic process, probably caused by two polling machines. These two machines determine the current operational status of network hosts by polling 300 machines per hour through gateway 6 and thus assure that even otherwise seldom used addresses are utilized at least once an hour.

Given the data above, an LRU cache should be effective for reducing the number of routing-table references. The probability of reference versus time of previous reference data allows simple calculation of an LRU cache hit ratio for a given number of cache slots. The relationship between cache hit ratio and probability of access is:

---

[2]For a cache large enough to hold entries for all simultaneous trains.

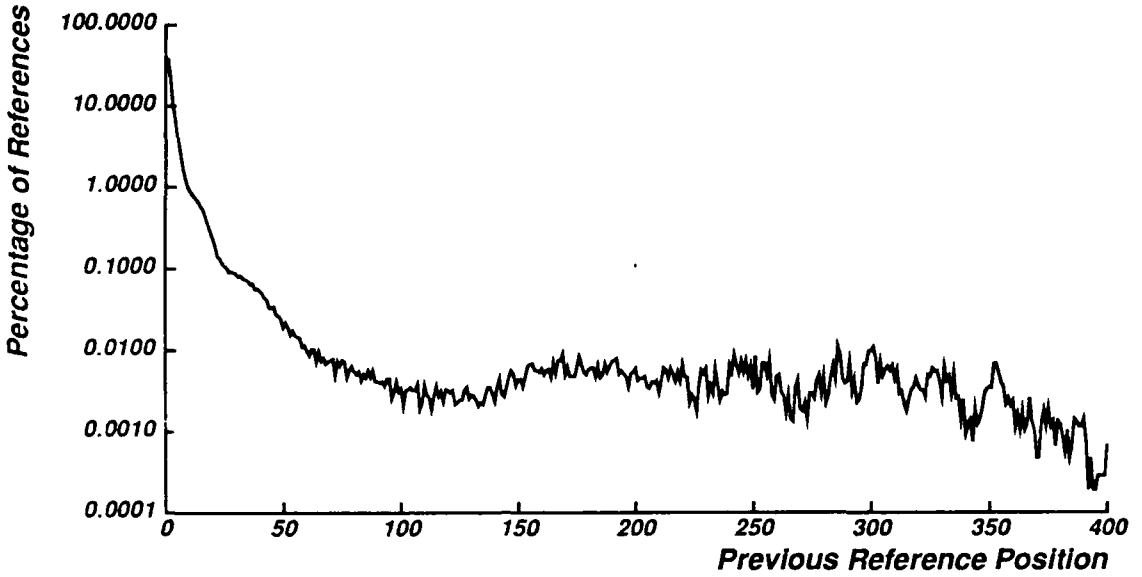


Figure 3-7: Percentage of Reference versus Time of Last Reference (Destination - Gateway 6)

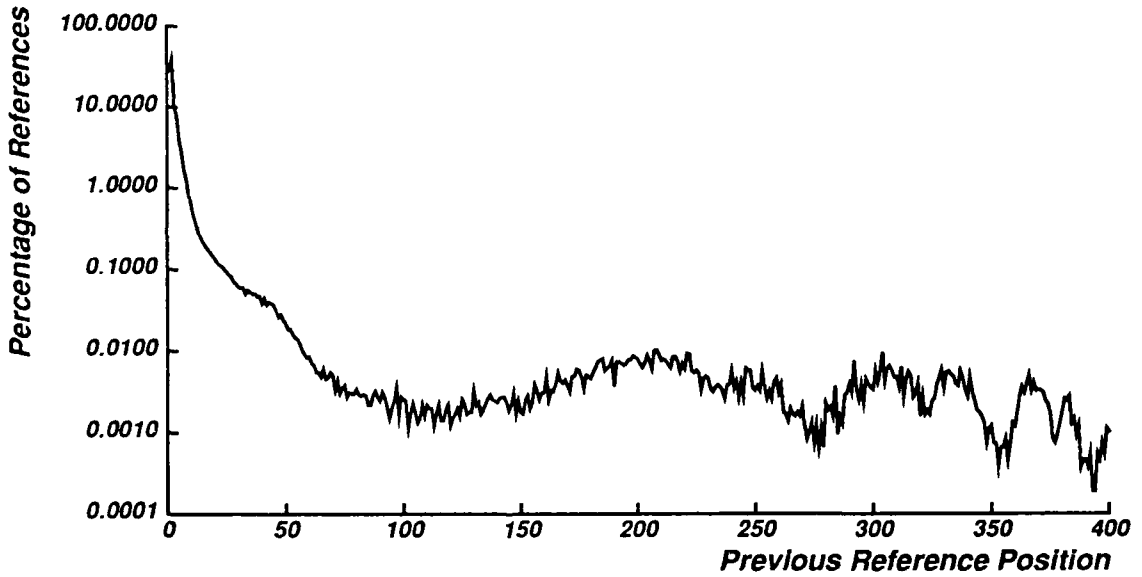


Figure 3-8: Percentage of Reference versus Time of Last Reference (Destination/Source - Gateway 6)

$$f_h(i) = \sum_{j=1}^{i} p_j$$

Where $f_h(i)$ is the cache hit ratio as a function of i, the number of cache slots, and p_j is the probability of a packet address being the j^{th} previous reference. Figures 3-9 and 3-10 show the percentage of cache hits versus cache size for both destination and destination/source caches gateways 4 and 6.

Tables 3-1 and 3-2 show just how quickly the probability of a cache hit climbs even for small LRU cache sizes. With 17 slots, the hit ratio is above 0.9 for all gateways. In fact, two gateways have a demand/prefetch cache hit ratio above 0.9 with only 6 slots. The small number listed for each gateway is its local network address. Underneath each gateway local address is the number of packets that the

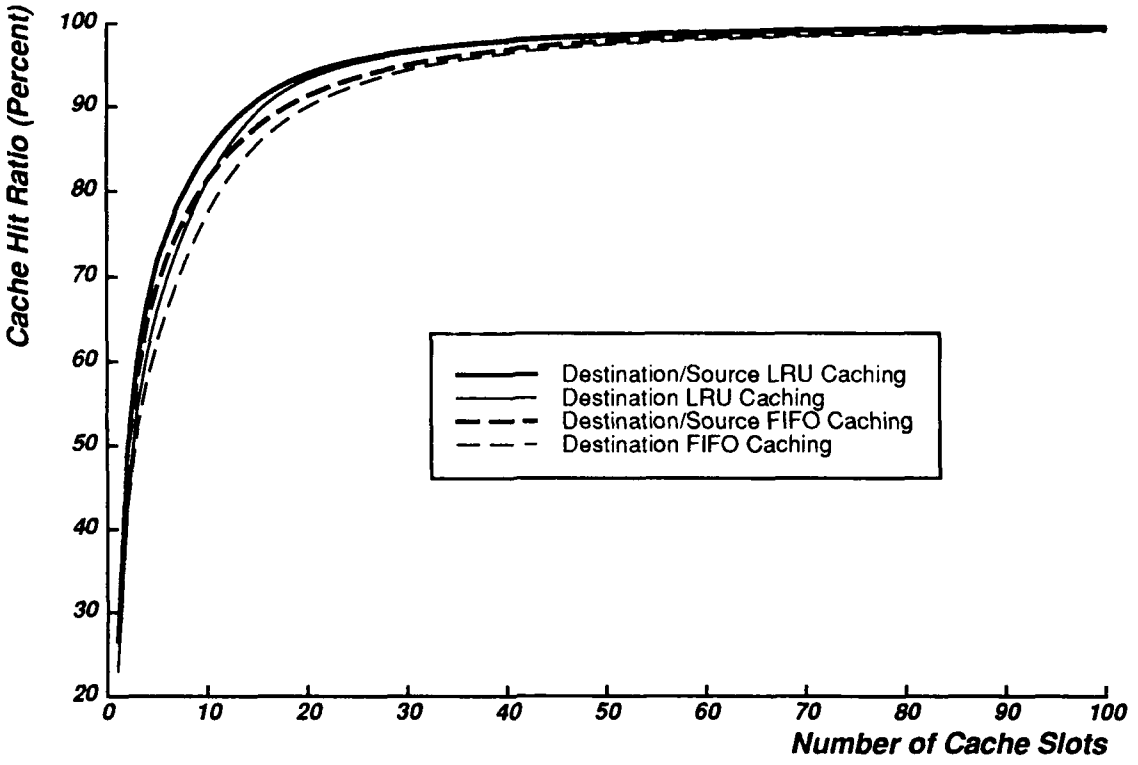


Figure 3-9: Percentage of Cache Hits versus Cache Size (Gateway 4)

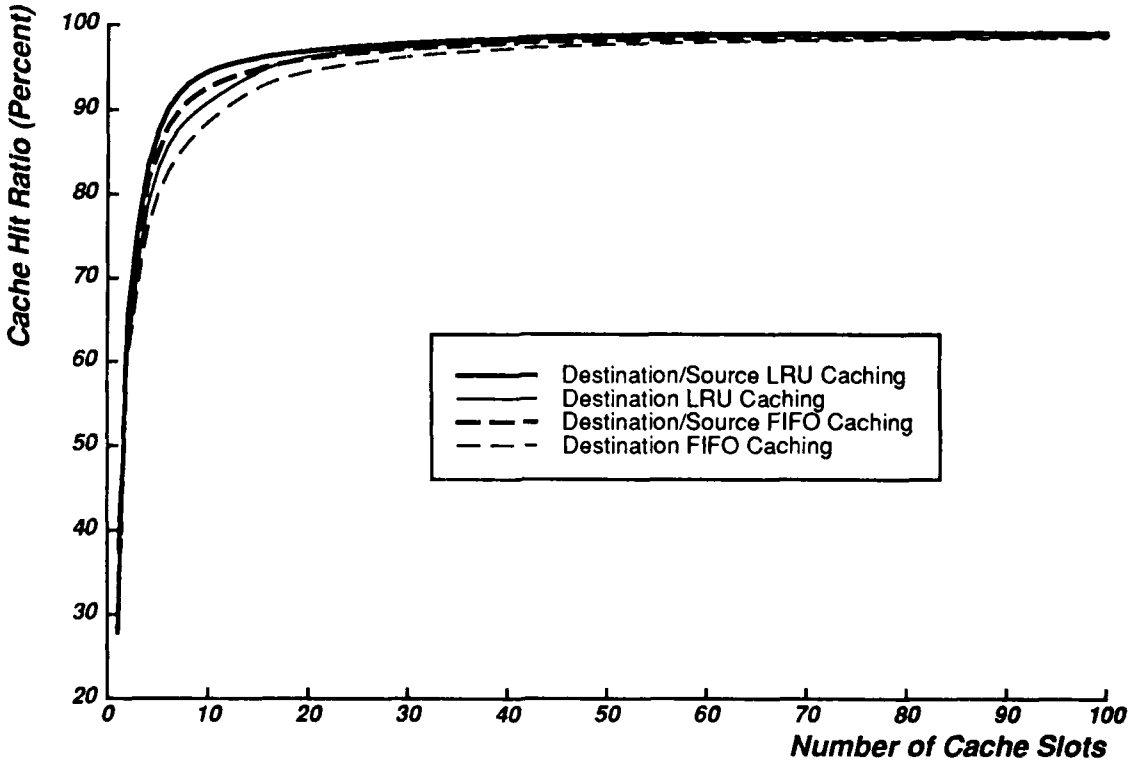


Figure 3-10: Percentage of Cache Hits versus Cache Size (Gateway 6)

gateway transported during the 24 hour measurement period. Tables 3-3 and 3-4 show the hit ratios for small FIFO caches.

Table 3-1: Table of Cache Hit Ratio Percentage versus Cache Size (Demand LRU)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	23.01	37.70	20.83	28.38	21.98
2	43.56	60.79	40.46	58.34	59.56
3	54.39	71.36	52.68	69.41	76.67
4	61.20	78.72	61.11	77.11	85.81
5	66.05	83.03	66.95	81.52	90.00
6	69.90	85.80	71.18	84.69	92.30
7	73.20	87.62	74.46	86.97	93.58
8	76.09	88.93	77.14	88.73	94.38
9	78.69	89.98	79.44	90.11	94.89
10	81.03	90.88	81.43	91.21	95.24
11	83.13	91.70	83.19	92.09	95.49
12	84.96	92.47	84.77	92.78	95.69
13	86.58	93.18	86.18	93.33	95.85
14	88.02	93.83	87.45	93.80	95.99
15	89.26	94.41	88.57	94.18	96.11
16	90.31	94.93	89.57	94.50	96.21
17	91.21	95.35	90.44	94.78	96.32
18	91.98	95.69	91.22	95.02	96.41
19	92.64	95.98	91.91	95.25	96.50
20	93.21	96.21	92.52	95.46	96.59

Table 3-2: Table of Cache Hit Ratio Percentage versus Cache Size (Demand/Prefetch LRU)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	26.57	27.86	27.40	37.58	46.44
2	49.58	65.56	48.22	65.96	68.43
3	60.42	75.82	62.13	74.43	83.39
4	66.93	83.44	68.08	80.68	89.08
5	71.94	87.30	74.18	83.26	92.40
6	75.09	90.07	77.01	86.16	94.20
7	78.16	91.76	80.18	87.69	95.08
8	80.47	93.03	82.18	89.45	95.73
9	82.65	93.85	84.14	90.61	96.10
10	84.42	94.48	85.63	91.79	96.38
11	86.04	94.94	87.01	92.66	96.56
12	87.41	95.30	88.15	93.44	96.69
13	88.64	95.61	89.17	94.05	96.79
14	89.69	95.86	90.04	94.57	96.87
15	90.64	96.07	90.81	94.99	96.94
16	91.45	96.27	91.50	95.34	97.00
17	92.16	96.44	92.11	95.62	97.06
18	92.79	96.61	92.65	95.86	97.11
19	93.34	96.75	93.12	96.07	97.16
20	93.83	96.89	93.54	96.24	97.21

Table 3-3: Table of Cache Hit Ratio Percentage versus Cache Size (Demand FIFO)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	23.01	37.70	20.83	28.38	21.98
2	43.02	59.86	40.33	57.74	59.38
3	51.95	69.42	50.59	67.69	74.88
4	57.91	76.03	57.75	74.64	83.29
5	62.42	80.00	62.89	78.94	87.63
6	66.16	82.72	66.86	82.01	90.20
7	69.44	84.66	70.16	84.31	91.75
8	72.36	86.16	72.92	86.13	92.75
9	74.98	87.42	75.35	87.59	93.45
10	77.31	88.53	77.50	88.81	93.93
11	79.36	89.52	79.39	89.82	94.30
12	81.19	90.43	81.08	90.65	94.61
13	82.80	91.26	82.58	91.33	94.86
14	84.21	91.97	83.91	91.89	95.07
15	85.46	92.58	85.10	92.37	95.25
16	86.54	93.10	86.15	92.78	95.41
17	87.51	93.53	87.08	93.15	95.55
18	88.39	93.89	87.93	93.47	95.68
19	89.17	94.20	88.69	93.77	95.80
20	89.87	94.47	89.37	94.04	95.90

Table 3-4: Table of Cache Hit Ratio Percentage versus Cache Size (Demand/Prefetch FIFO)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	26.57	27.86	27.40	37.58	46.44
2	49.49	65.08	48.20	65.43	68.55
3	58.14	73.59	59.50	72.29	81.06
4	64.74	81.55	65.79	78.82	87.72
5	68.88	84.95	70.64	81.46	90.82
6	72.30	87.99	73.99	84.28	92.90
7	75.03	89.65	76.80	85.99	93.98
8	77.41	91.02	79.02	87.62	94.74
9	79.43	91.90	80.91	88.85	95.23
10	81.22	92.65	82.52	89.93	95.60
11	82.78	93.21	83.90	90.81	95.86
12	84.19	93.69	85.11	91.58	96.05
13	85.44	94.11	86.16	92.26	96.21
14	86.54	94.46	87.10	92.84	96.34
15	87.54	94.80	87.95	93.33	96.46
16	88.44	95.09	88.71	93.79	96.56
17	89.23	95.36	89.38	94.16	96.65
18	89.95	95.60	90.01	94.50	96.72
19	90.59	95.80	90.57	94.79	96.79
20	91.18	96.00	91.09	95.02	96.85

Notice that for the LCS Ethernet gateway, a single-slot cache works better without prefetching, unlike the other gateways. This implies that a packet from A to B through gateway 6 is more likely to be followed by another in the same direction, rather than a packet returning in the opposite direction. This behavior is characteristic of a file transfer and a popular application that operates through this gateway is Remote Virtual Disk (RVD) file transfers.

Compared with the results in an earlier paper [4], the hit ratio of the ARPANET gateway is somewhat lower in this set of measurements. One of the reasons is that more distinct addresses were seen during the current measurement period (1841 versus 1035 destination addresses, 2019 versus 1130 destination/source addresses). Another reason is that the ARPANET gateway was part of a packet-forwarding loop with gateway 7. For each packet that arrived at the ARPANET gateway with a certain set of addresses, the ARPANET gateway would forward the packet to gateway 7 and vice-versa, each packet making 256 hops before being killed, thus increasing the packet address locality seen at the ARPANET gateway. Each of 11 addresses was seen about 37,000 times during the measurement period, for a total of 408,655 packets out of 2,147,956 (19%).

Another way of looking at the hit ratio data is to plot the average number of packets through the gateway between cache misses. The relation between cache hit ratio and packets between cache misses is:

$$f_m(i) = \frac{1}{1 - f_h(i)}$$

Where $f_m(i)$ is the number of packets between cache misses as a function of i, the number of cache slots, and $f_h(i)$ is the probability of a cache hit as a function of the number of cache slots.

The effectiveness of a cache slot depends on the ratio of the packets-between-misses values before and after the slot, not an absolute number, so the graphs in figures 3-11 and 3-12 are plotted on a logarithmic scale so that equal ratios appear as equal vertical intervals. This graph shows the incremental efficiency of each additional slot in the cache. All graphs contain curves for destination LRU, destination/source LRU, destination FIFO, and destination/source FIFO in increments of 1, except for the ARPANET gateway FIFO curves, which are in increments of 10. In addition, the graph for the ARPANET gateway also shows the destination and destination/source curves for a two cache system consisting of a 100 slot LRU cache followed by a FIFO cache. The LRU portion of the curve is in increments of 1 and the FIFO portion of the curve is in increments of 2.

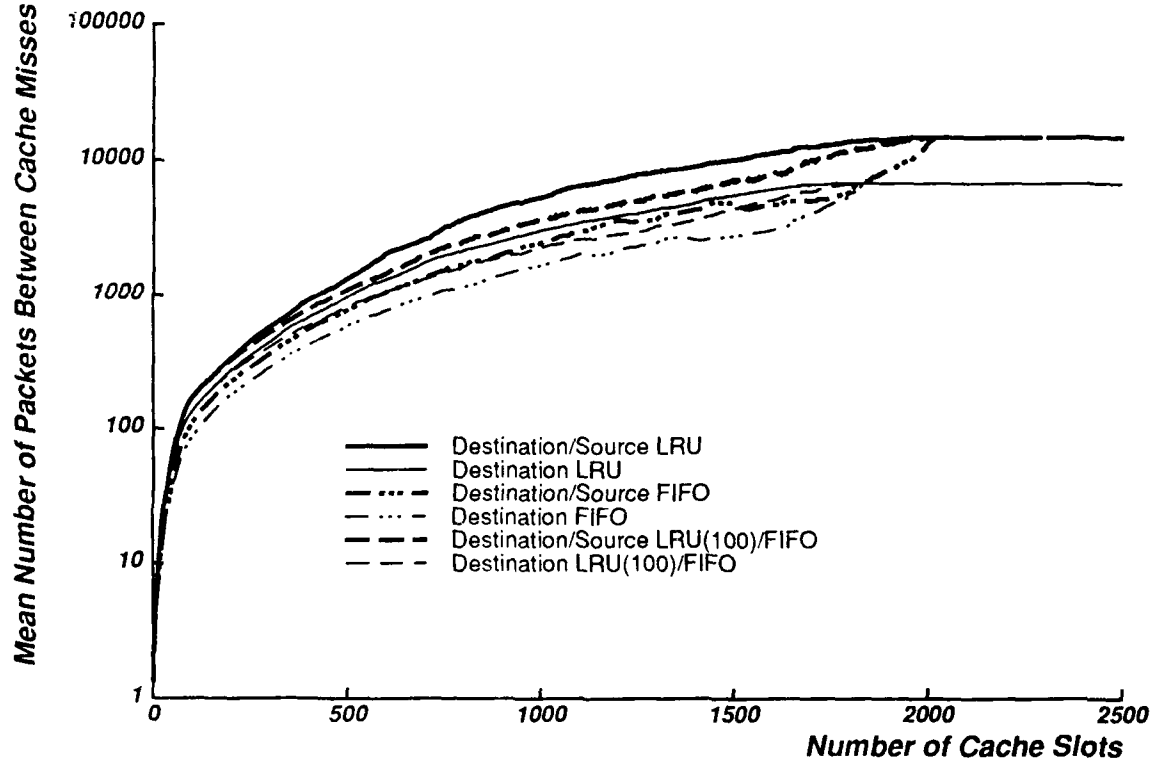


Figure 3-11: Number of Packets Between Cache Misses versus Cache Size (Log-Linear - Gateway 4)

As expected, the destination/source LRU cache performed best, and each destination/source cache outperforms the corresponding destination only cache, because much of the traffic through the gateways is bidirectional. When the caches become large enough to contain all addresses seen by the gateway for the measurement period, all the destination caches converge and all the destination/source caches converge to the same values.

The incremental efficiency of each slot in the cache is the slope of the curve at that slot number. A sharp decrease in the slope indicate a point of diminishing returns for larger cache sizes. Particularly

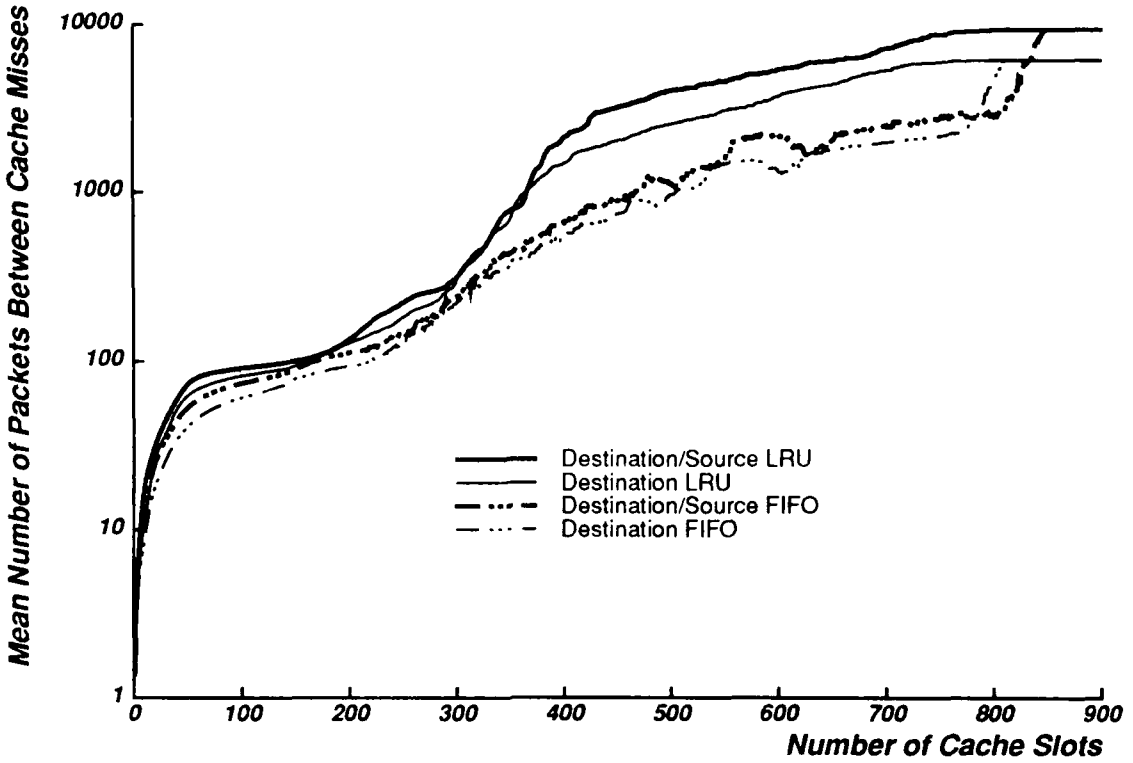


Figure 3-12: Number of Packets Between Cache Misses versus Cache Size (Log-Linear - Gateway 6)

notice the drop in LRU cache efficiency around cache slot 100 for the ARPANET gateway and slot 50 for the LCS Ethernet gateway.

For both graphs, the number of packets between cache misses is not a monotonically increasing function of cache size for FIFO caches. A similar type of anomaly was detected and explained for FIFO page replacement in computer systems with virtual memory [1].

Notice that for gateway 6, the cache performance increases sharply for cache sizes exceeding 250 slots. This anomaly is caused by polling machines that determine what machines on the network are in operation. One polling machine polls the 130 machines on the LCS Ethernet through gateway 6. Another polling machine on the LCS Ethernet polls 170 machines on networks other than the LCS Ethernet. Both of these machines poll a total of 300 machines through gateway 6 once an hour. These two machines together cause the cache performance to be lower than it should for cache sizes less than 300 because polling exhibits low address locality.

If the interest is in determining how many cache slots are necessary for a given number of packets between cache misses, the above graph is better plotted on a linear-linear scale; see figures 3-13 and 3-14. From these plots, we see that prefetching is more important for gateway 4 than it is for gateway 6; this is because gateway 4 carries more bidirectional traffic than gateway 6.

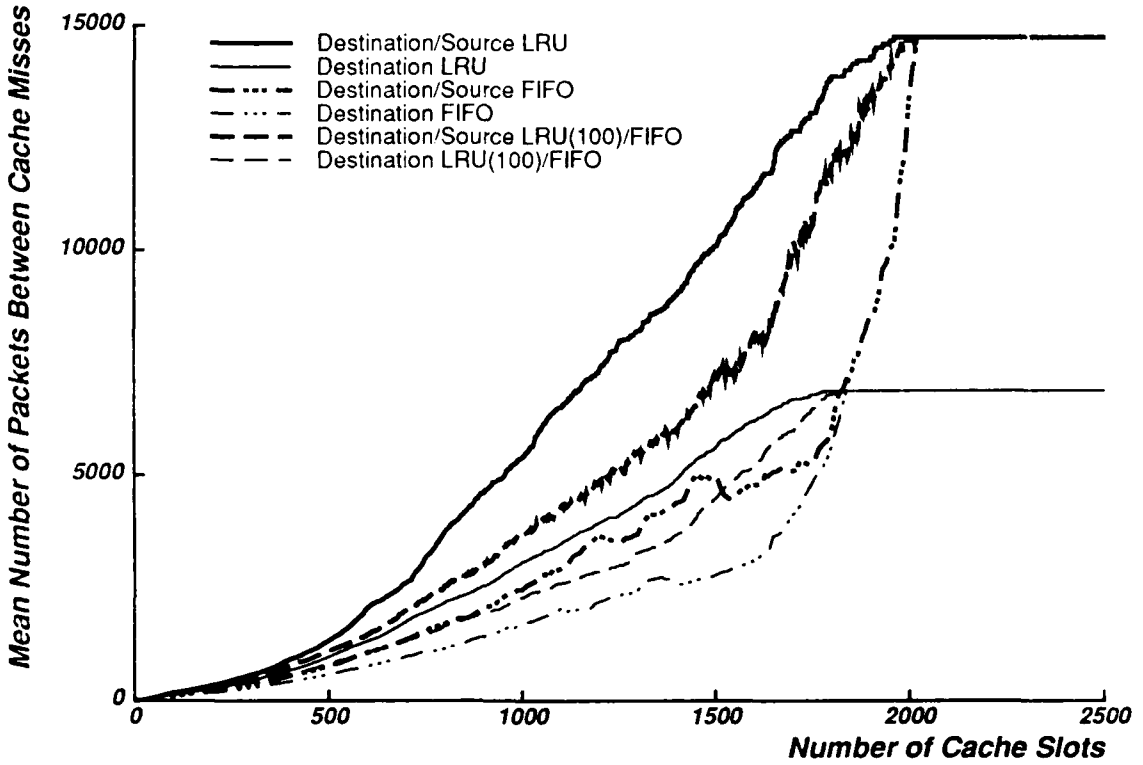


Figure 3-13: Number of Packets Between Cache Misses versus Cache Size (Linear-Linear - Gateway 4)

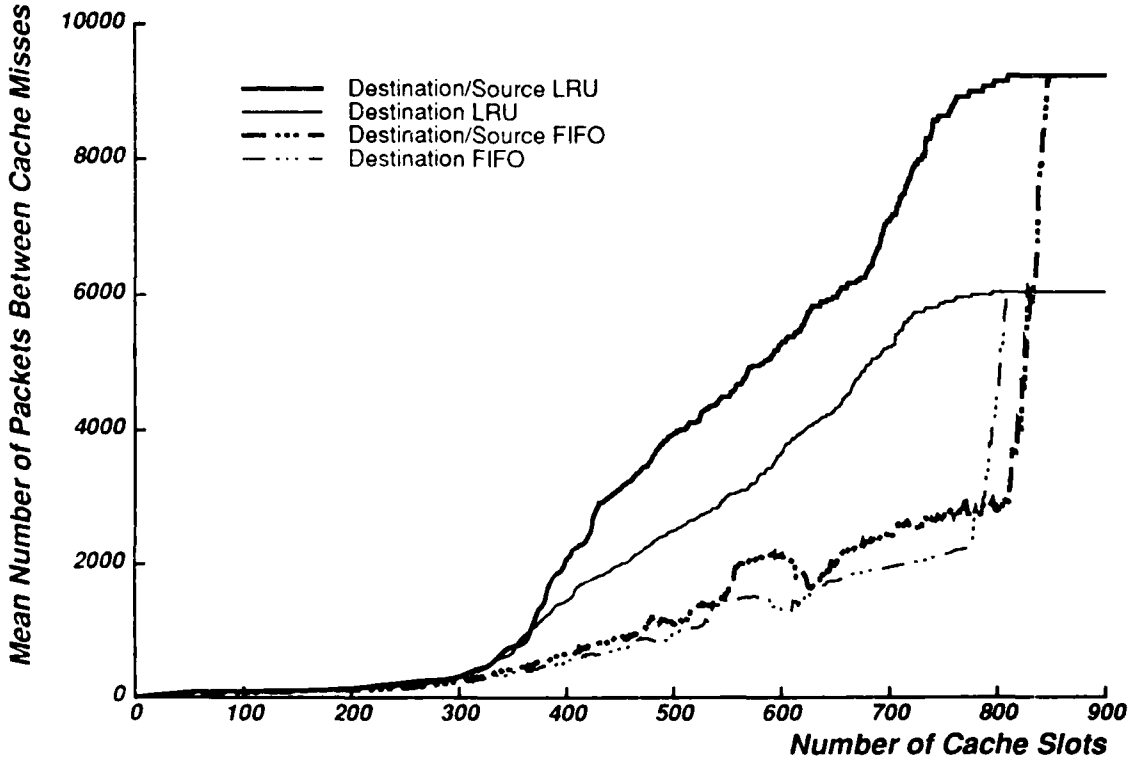


Figure 3-14: Number of Packets Between Cache Misses versus Cache Size (Linear-Linear - Gateway 6)

3.2. Results for Hierarchical Addressing

Although in the previous section packet addresses were treated as flat addresses, in reality they are hierarchical addresses, which means that an address can be separated into network and host pieces. Currently, routing is done based on network addresses and since the number of networks is smaller than the number of hosts, the cache hit ratio for a network cache is higher than the hit ratio of a flat address cache. In order to cache network addresses, the complete address must pass through an address decoder that extracts the network address. The tradeoff is that the hit ratio of the cache is higher than for flat addresses, but each address must be decoded.

Hierarchical addressing, as done by current gateways, is complex because the separation of the network and host fields of the internet address is determined by the type of address. In order to avoid this complication, the separation of fields for the purposes of this research is that the first three bytes of the full address refer to the network and that the last byte refers to the host. This procedure never removes part of the network address as long as the address is not a class C address with subnetting (which is not known to be used by anyone). In some cases, some of the host's address will be left with the network address, which means that the cache hit ratio will be lower than if the address were completely separated.

Since the purpose of a cache is to speed up routing table access, it is probably advantageous to place the cache before the address separation, rather than after it, so that each cache hit saves not only a table lookup, but also the trouble of dissecting the full address. It may be advantageous to use a simple address separation procedure before the cache, such as the suggested one of removing the last byte of the address. This preprocessing step increases the cache hit ratio by eliminating most, if not all, of the host address from the full address, and yet avoids the complexity of complete address decoding.

Figure 3-15 shows the percentage of references versus time of previous reference for the 200 most recently referenced packet destination addresses for the ARPANET gateway. Figure 3-16 is identical to figure 3-15 except that both packet destination and source addresses are used. The same information for 70 previous references for the LCS Ethernet gateway is seen in figures 3-17 and 3-18. All graphs are plotted on a logarithmic scale so that the relative probabilities are more easily seen.

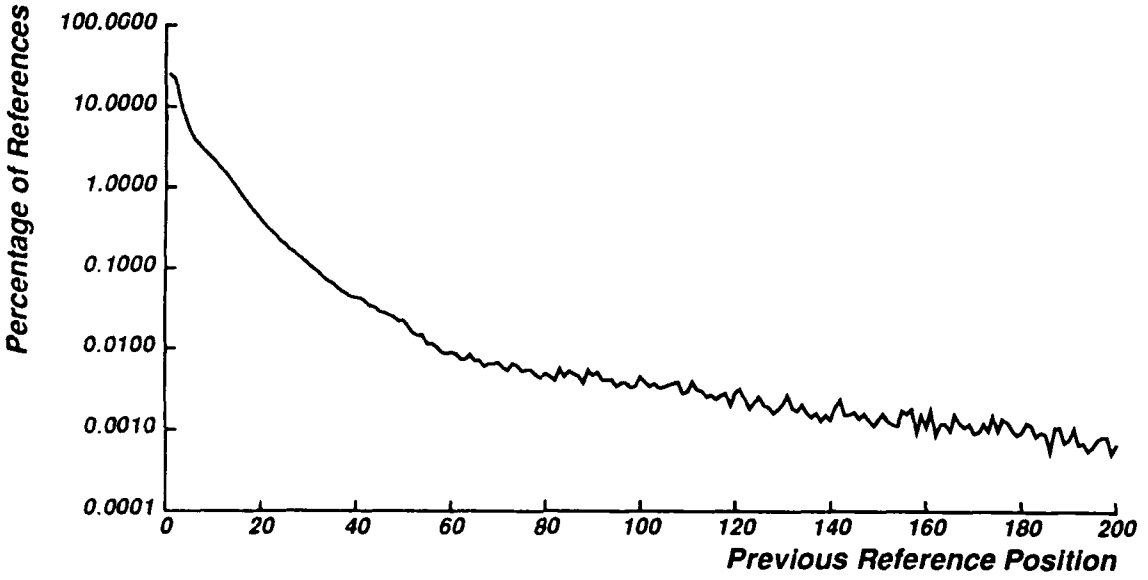


Figure 3-15: Percentage of Reference versus Time of Last Reference (Destination - Gateway 4)

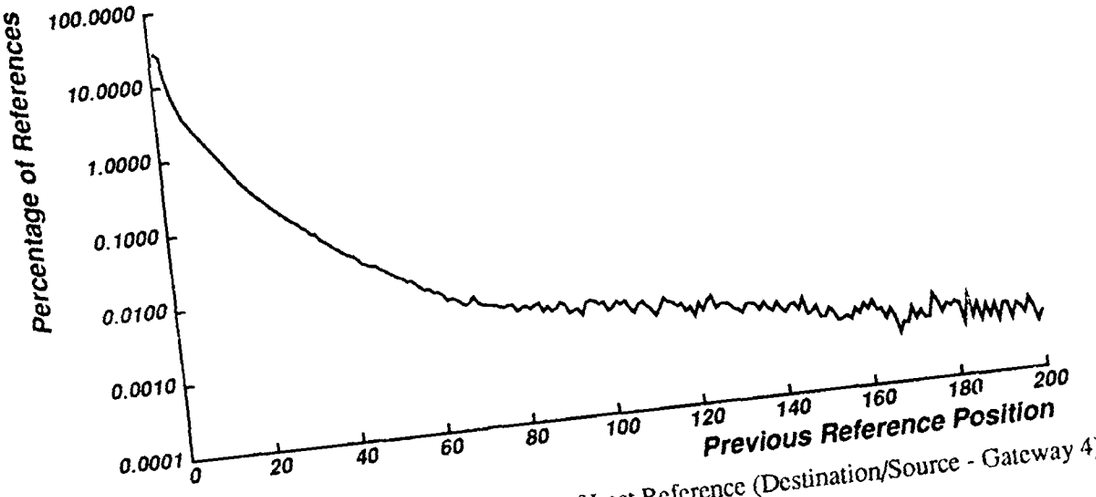


Figure 3-16: Percentage of Reference versus Time of Last Reference (Destination/Source - Gateway 4)

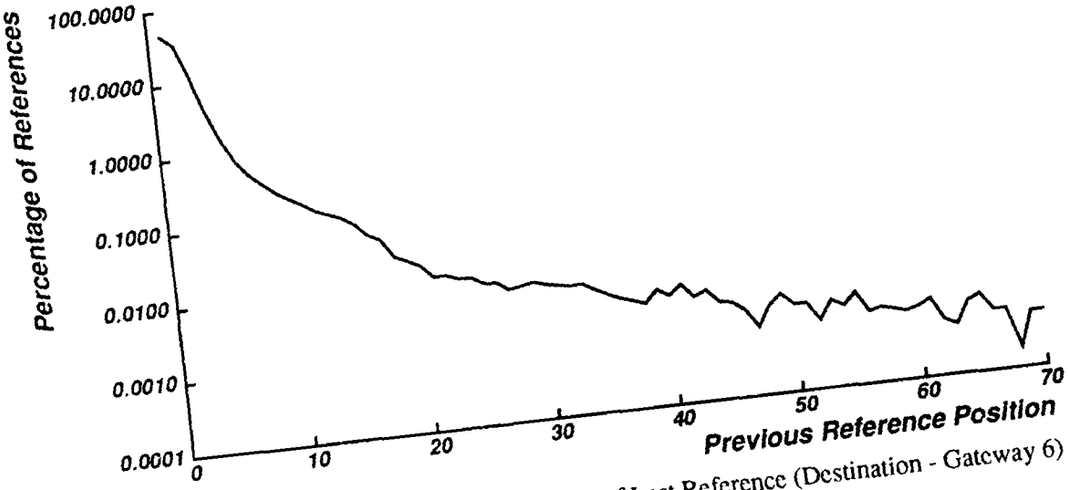


Figure 3-17: Percentage of Reference versus Time of Last Reference (Destination - Gateway 6)

Figures 3-19 and 3-20 show the percentage of cache hits versus cache size for both destination and destination/source LRU and FIFO caches. Notice that even a relatively small cache has a high cache hit ratio.

Tables 3-5 and 3-6 show just how quickly the probability of a cache hit climbs even for small LRU cache sizes. With 13 slots, the hit ratio is already above 0.9 for all gateways. In fact, two gateways have a demand cache hit ratio above 0.9 with only 3 slots and above 0.99 with 9 slots. Tables 3-7 and 3-8 show the FIFO cache hit ratio for small caches.

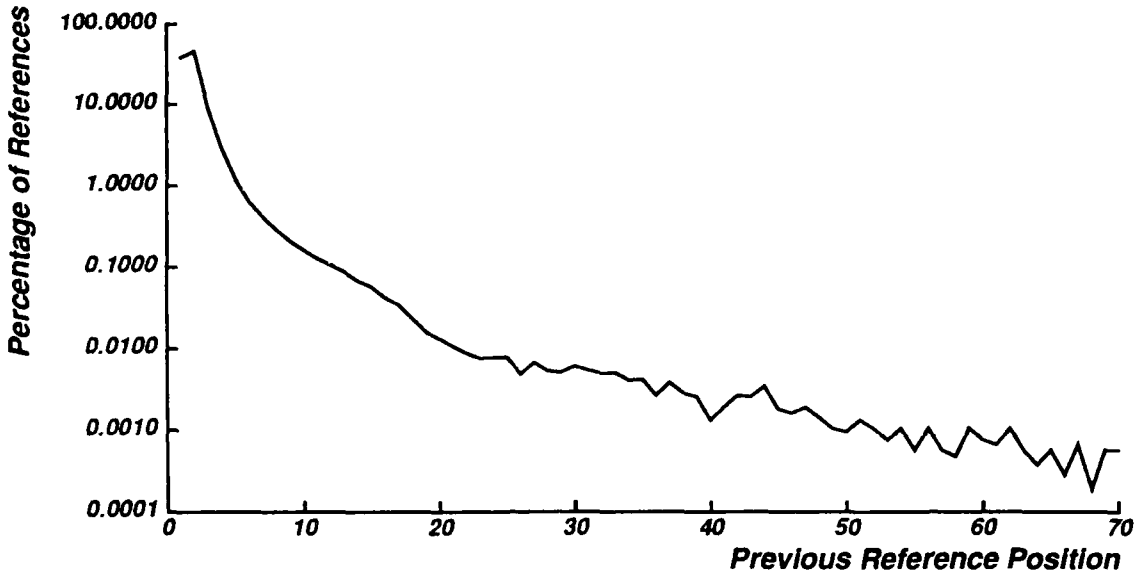


Figure 3-18: Percentage of Reference versus Time of Last Reference (Destination/Source - Gateway 6)

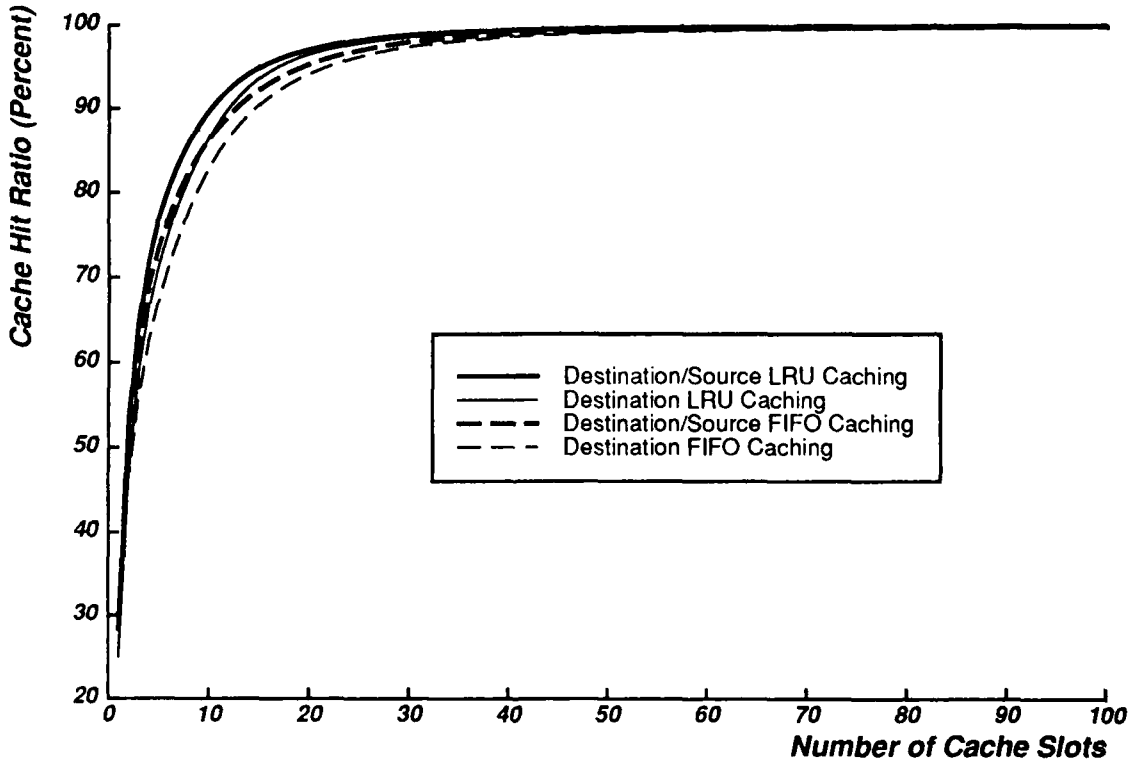


Figure 3-19: Percentage of Cache Hits versus Cache Size (Gateway 4)

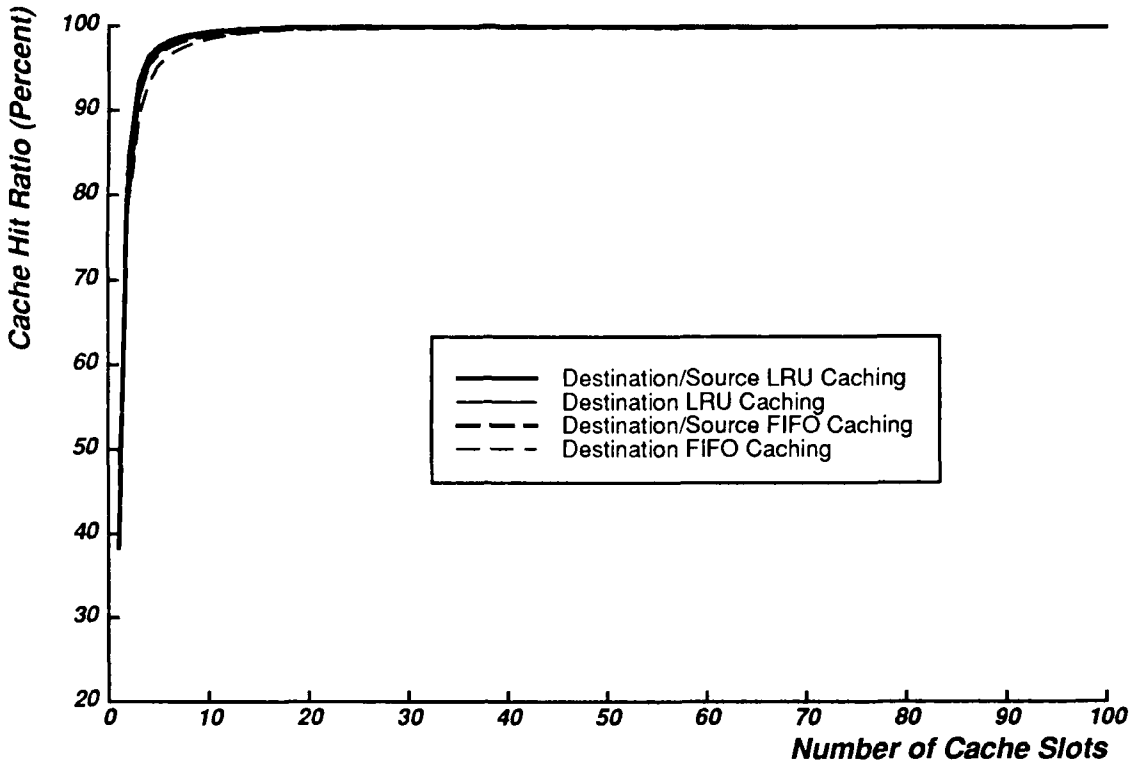


Figure 3-20: Percentage of Cache Hits versus Cache Size (Gateway 6)

Table 3-5: Table of Cache Hit Ratio Percentage versus Cache Size (Demand LRU)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	25.01	45.95	22.83	33.47	29.35
2	47.41	79.35	45.18	66.55	79.56
3	59.28	91.44	57.99	79.70	92.29
4	66.54	95.39	66.58	87.13	96.08
5	71.58	96.94	72.36	91.33	97.56
6	75.53	97.72	76.50	94.07	98.23
7	78.86	98.22	79.71	95.94	98.60
8	81.76	98.57	82.34	97.25	98.85
9	84.30	98.83	84.59	98.14	99.04
10	86.52	99.04	86.56	98.76	99.18
11	88.44	99.21	88.29	99.16	99.30
12	90.08	99.34	89.83	99.42	99.40
13	91.46	99.46	91.18	99.58	99.48
14	92.62	99.56	92.34	99.67	99.55
15	93.58	99.63	93.34	99.73	99.60
16	94.37	99.69	94.19	99.77	99.64
17	95.04	99.73	94.92	99.80	99.68
18	95.59	99.76	95.54	99.82	99.71
19	96.07	99.78	96.07	99.84	99.73
20	96.47	99.80	96.52	99.86	99.76

Table 3-6: Table of Cache Hit Ratio Percentage versus Cache Size (Demand/Prefetch LRU)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	28.31	38.39	29.95	39.77	56.23
2	53.32	84.34	52.78	73.24	85.58
3	64.86	93.50	66.89	81.89	93.80
4	72.04	96.36	73.42	89.21	96.82
5	77.07	97.55	79.03	91.90	98.08
6	80.62	98.19	82.07	94.27	98.65
7	83.58	98.61	84.93	95.72	98.96
8	86.00	98.90	86.97	96.79	99.15
9	88.02	99.11	88.75	97.56	99.30
10	89.72	99.27	90.19	98.15	99.41
11	91.13	99.39	91.42	98.61	99.49
12	92.33	99.50	92.45	98.95	99.56
13	93.32	99.59	93.33	99.20	99.62
14	94.15	99.66	94.10	99.39	99.66
15	94.85	99.71	94.75	99.52	99.70
16	95.43	99.76	95.32	99.61	99.73
17	95.93	99.79	95.80	99.67	99.76
18	96.35	99.81	96.24	99.72	99.78
19	96.71	99.83	96.62	99.75	99.80
20	97.03	99.84	96.95	99.78	99.81

Table 3-7: Table of Cache Hit Ratio Percentage versus Cache Size (Demand FIFO)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	25.01	45.95	22.83	33.47	29.35
2	46.58	78.39	44.84	65.78	78.83
3	56.35	89.32	55.40	77.75	90.25
4	62.75	93.43	62.77	84.72	94.49
5	67.56	95.41	68.01	89.03	96.35
6	71.54	96.52	72.07	91.98	97.31
7	74.97	97.24	75.42	94.15	97.88
8	77.98	97.77	78.26	95.77	98.27
9	80.62	98.16	80.77	96.96	98.57
10	82.92	98.48	82.96	97.79	98.78
11	84.90	98.72	84.89	98.41	98.96
12	86.62	98.93	86.57	98.81	99.09
13	88.06	99.09	88.04	99.08	99.19
14	89.34	99.22	89.28	99.26	99.29
15	90.42	99.32	90.39	99.39	99.35
16	91.37	99.40	91.37	99.49	99.42
17	92.20	99.47	92.23	99.56	99.47
18	92.93	99.52	92.98	99.62	99.52
19	93.56	99.57	93.65	99.67	99.55
20	94.12	99.60	94.25	99.71	99.59

Table 3-8: Table of Cache Hit Ratio Percentage versus Cache Size (Demand/Prefetch FIFO)

Number of Cache Slots	Gateway 4 2,847,682	Gateway 6 1,062,294	Gateway 7 3,769,544	Gateway 9 914,570	Gateway 11 1,067,433
1	28.31	38.39	29.95	39.77	56.23
2	52.90	83.52	52.56	71.62	84.54
3	62.32	92.57	63.95	79.50	92.66
4	69.25	95.48	70.60	86.18	96.02
5	73.70	96.82	75.34	89.48	97.46
6	77.25	97.59	78.76	91.95	98.18
7	80.12	98.09	81.50	93.71	98.59
8	82.54	98.45	83.74	95.09	98.85
9	84.57	98.72	85.61	96.13	99.05
10	86.34	98.92	87.17	96.96	99.19
11	87.83	99.09	88.52	97.62	99.30
12	89.17	99.23	89.68	98.11	99.39
13	90.35	99.33	90.70	98.49	99.46
14	91.35	99.42	91.61	98.77	99.52
15	92.23	99.50	92.41	98.98	99.56
16	93.01	99.56	93.13	99.14	99.60
17	93.68	99.60	93.76	99.27	99.64
18	94.28	99.64	94.33	99.37	99.66
19	94.80	99.67	94.83	99.43	99.69
20	95.27	99.70	95.29	99.49	99.72

Notice that the demand/prefetch cache hit ratio for a 1-slot LRU cache for gateway 6 is less than the demand-only hit ratio. This is an example of a case where caching the packet source address along with the packet destination address does not improve the cache hit ratio.

Another way of looking at this data is to plot the average number of packets through the gateway between cache misses. The graph in figure 3-21 shows the incremental efficiency of each additional slot in the cache. Curves are shown for destination LRU, destination/source LRU, destination FIFO, and destination/source FIFO.

As expected, the destination/source LRU cache performed best, followed closely by the destination LRU cache. The graph for gateway 6 is similar. The same graphs plotted on a linear-linear scale are shown in figures 3-23 and 3-24.

As can be seen from the above data, hierarchical address caching is better than flat address caching. The improvement in cache hit ratio for 10 element LRU caches ranges from 3% to 8% and the improvement in the number-of-packets-between-cache-misses ratio ranges from 2.9 to 4.3 times better for the maximum size caches for gateways 4 and 6.

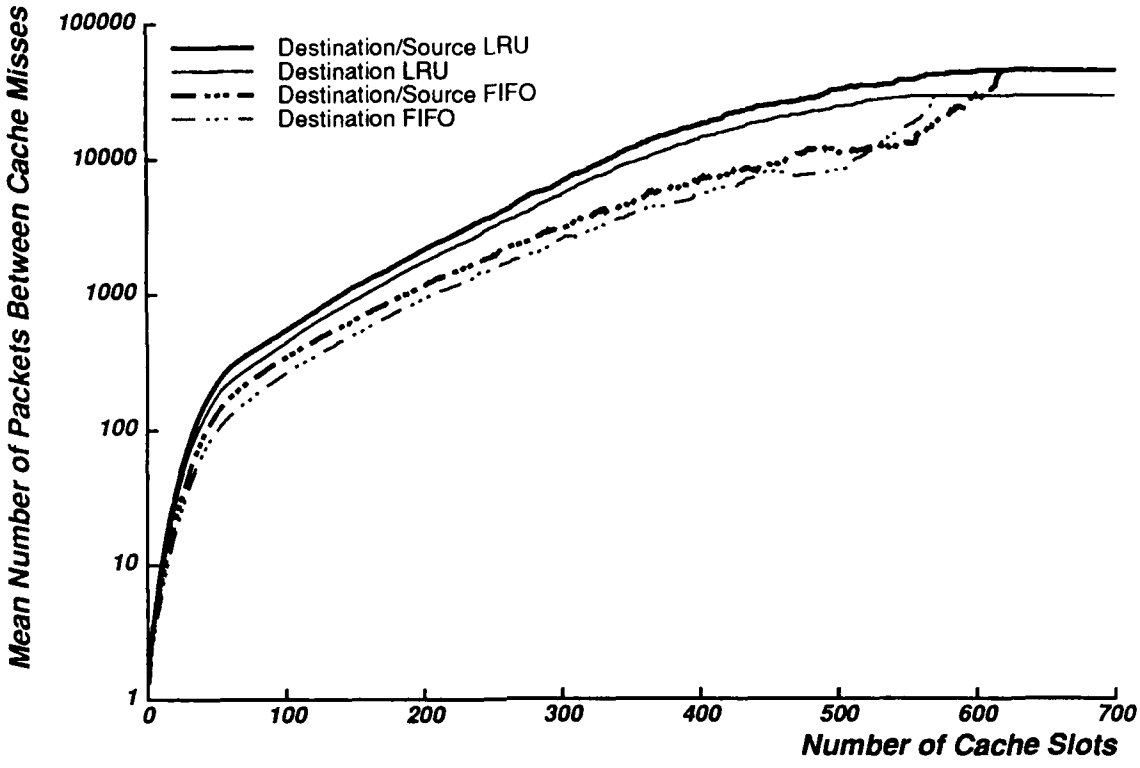


Figure 3-21: Number of Packets Between Cache Misses versus Cache Size (Log-Linear - Gateway 4)

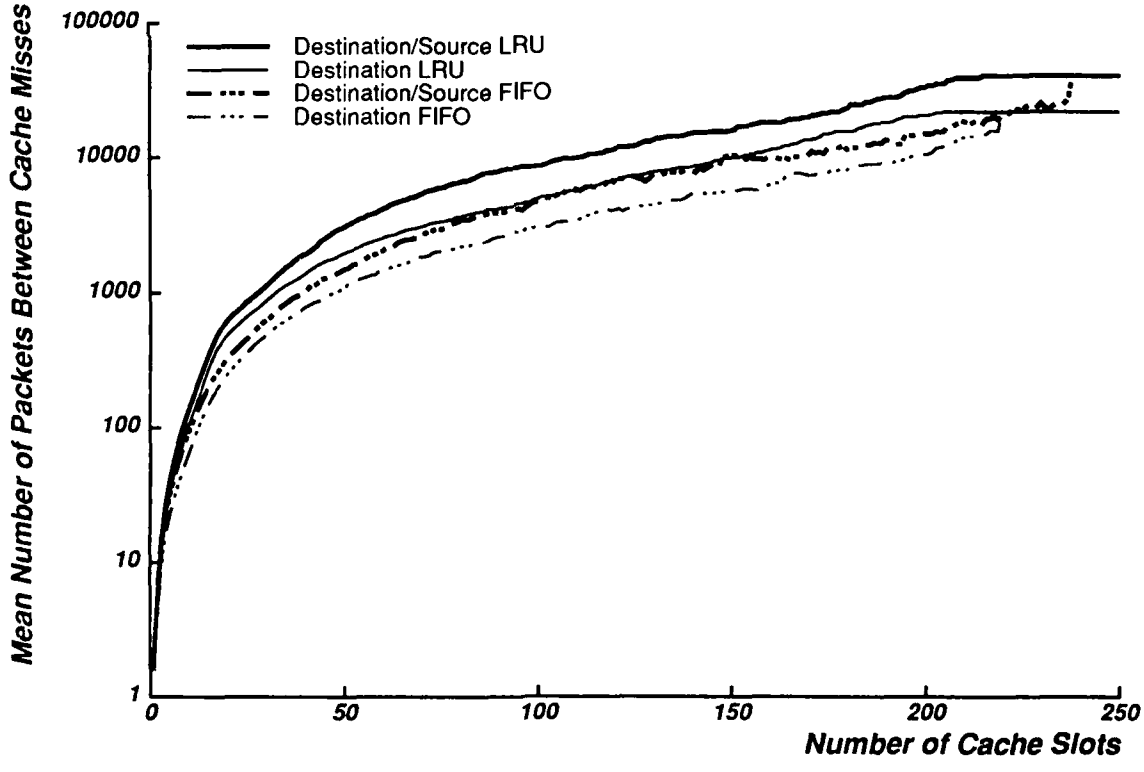


Figure 3-22: Number of Packets Between Cache Misses versus Cache Size (Log-Linear - Gateway 6)

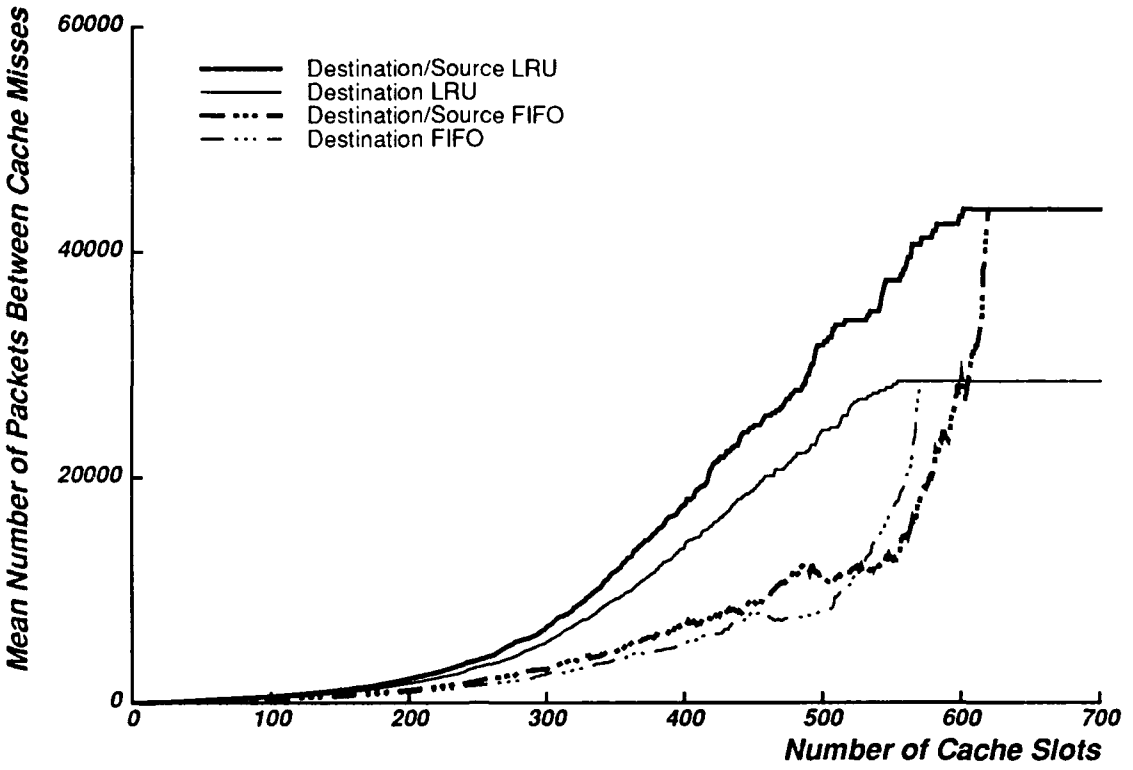


Figure 3-23: Number of Packets Between Cache Misses versus Cache Size (Linear-Linear - Gateway 4)

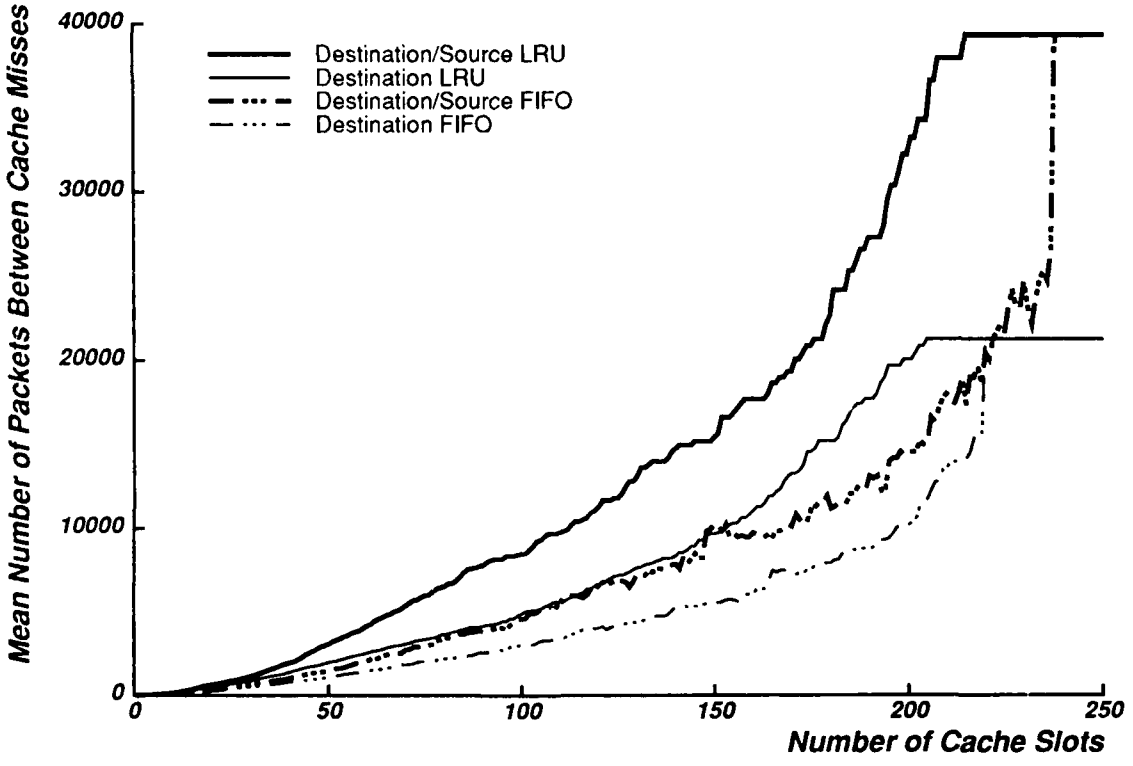


Figure 3-24: Number of Packets Between Cache Misses versus Cache Size (Linear-Linear - Gateway 6)

4. Caching Cost Analysis

Although the hit ratios have been given for various gateway caches, the important question is what is the performance improvement that is gained for the gateway. The following analysis is meant to be a simple worst-case analysis. The cache used for this analysis is a fully associative cache with an LRU replacement policy. The cache is implemented completely in software as a doubly-linked linear list. The list is searched linearly from most recently used to least recently used and when the list is rearranged to preserve the LRU ordering, this is done by swapping pointers of the doubly-linked list. Only destination addresses are cached.

Estimates of the number of instructions are based on a load-store machine architecture. Loading the pointer to the cache requires 1 instruction. A matching entry is recognized in 3 instructions; in addition, 7 instructions are needed for each previously checked cache slot After each cache search, the cache must be reordered to preserve the LRU ordering (except if the entry was found in the first slot) and this takes 21 instructions (except if the entry is in the last cache slot, which requires 11 instructions).

An estimate of the minimum-time address decode and routing table lookup for a packet is 80 instructions, a figure obtained by estimating the number of load-store instructions generated by the C gateway code at MIT. The current gateway code would take even longer; this estimate is conservative because it does not include function call overhead, error handling, or the additional routing table lookup necessary for subnet routing and assumes that there are no hashing collisions. This estimate also does not include the time necessary to map a next-hop IP address into a local network address, which is at least 2 instructions, but varies depending on the local network.

The average lookup time with an LRU cache is now:

$$\begin{aligned} & 1 + 3p_1 + (7+3+21)p_2 + (14+3+21)p_3 \\ & \quad + \cdots + (7[k-1]+3+21)p_k \\ & \quad + \cdots + (7[n-1]+3+11)p_n \\ & \quad + \Big(1 - \sum_{i=1}^{n} p_i\Big)(7n+21+80) \end{aligned}$$

If the cache stores flat addresses and destinations only, then the optimum number of cache slots for the ARPANET gateway is 18. The average lookup time with the cache becomes 60.0 instructions, 75% as long as a table lookup. For gateway 6, the optimal cache size is 16 slots and the average lookup time is 39.5 instructions, 49% as long as table lookup. With hierarchical addressing, the performance is even better. To obtain a network address from a flat address, the last byte is dropped from the address. Although this does not necessarily result in a network address, the number of distinct addresses that the cache must handle is reduced and the overhead is only a single AND instruction. For the ARPANET gateway, the optimal cache size is 24 slots and the average lookup time is 50.6 instructions, 63% as long as table lookup. For gateway 6, the optimal cache size is 19 slots and the average lookup time is 23.7 instructions, 30% as long as table lookup. These results are summarized in table 4-1

Table 4-1: Table of Optimal-Length LRU Cache Performance

Gateway	addressing method	number of slots	mean number of instructions	percentage of RT lookup
Gateway 4	flat	18	60.0	75%
Gateway 4	hier	24	50.6	63%
Gateway 6	flat	16	39.5	49%
Gateway 6	hier	19	23.7	30%

Although the figures above minimize average lookup time, it may be that a lower maximum lookup time is desired at the cost of a slightly higher average lookup time.

One problem with LRU caches is that it is expensive to maintain LRU ordering. Perhaps we can do better by having a simple cache that resides in the registers of the processor. The time for a single element cache where the cache uses 1 processor register is:

$$2p_1 + 83(1-p_1)$$

For hierarchical caches, add 1 to the above for an additional AND instruction. The results are presented in table 4-2.

Table 4-2: Table of Single-Slot Cache Performance

Gateway	addressing method	fetch method	mean number of instructions	percentage of RT lookup
Gateway 4	flat	dest	64.4	80%
Gateway 4	flat	d/s	61.5	77%
Gateway 4	hier	dest	62.2	78%
Gateway 4	hier	d/s	59.6	74%
Gateway 6	flat	dest	52.5	66%
Gateway 6	flat	d/s	59.9	75%
Gateway 6	hier	dest	44.9	56%
Gateway 6	hier	d/s	51.1	64%

A two-element cache requires two processor registers and the expected lookup time is:

$$2p_1 + 7p_2 + 86(1-p_1-p_2)$$

For dest/source caches:

$$4p_1 + 5p_2 + 86(1-p_1-p_2)$$

For hierarchical caches, add 1 to the above for an additional AND instruction. The results are presented

in table 4-3.

Table 4-3: Table of Dual-Slot Cache Performance

Gateway	addressing method	fetch method	mean number of instructions	percentage of RT lookup
Gateway 4	flat	dest	50.4	63%
Gateway 4	flat	d/s	45.6	57%
Gateway 4	hier	dest	48.3	60%
Gateway 4	hier	d/s	43.5	54%
Gateway 6	flat	dest	36.1	45%
Gateway 6	flat	d/s	32.5	41%
Gateway 6	hier	dest	22.0	28%
Gateway 6	hier	d/s	18.2	23%

Obviously, a better cache implementation would increase performance. Cache lookup could use a hash table rather than a linear search, the cache slots need not be fully associative, a replacement strategy approximating LRU may be implemented less expensively, the cache fetch strategy could use both destination and source addresses, or a hardware cache could be built into the system. But even with this simple cache implementation and optimistic assumptions about routing-table lookup time, the best cache above reduces lookup time by 77%.

5. Conclusion

Processing time per packet is reduced if average routing-table access time is reduced, and an economical way to reduce access time is to use a cache. Recordings were made of routing-table accesses for several operating gateways and these records were used to drive cache simulations to determine the hit ratio for various types of caches. The caches simulated were fully-associative, LRU or FIFO, and cached destination addresses or destination and source addresses. Results were computed for both flat and hierarchical addressing schemes.

The probability of reference to a destination address versus time of previous reference to that address is monotonically decreasing for up to 100 previous references, implying that an LRU cache management procedure is optimal for caches of 100 slots or less.

Locality of packet flat addresses causes an LRU cache of less than 20 slots to have a hit ratio of over 90% for all measured gateways; two gateways need only 6 slots to exceed a 90% hit ratio. Hierarchical addressing measurements place all gateway hit ratios at over 90% for 13 slots and 2 gateways are over 90% with only 3 slots and over 99% with 9 slots.

In addition to caching the packet destination/next-hop pair, it is worth caching the packet source/next-hop pair if it can be done inexpensively. In some cases, this can be detrimental for a small cache because the turn-around time for response packets on a slow network could mean that a source address is expelled form the cache before the corresponding packet from the opposite direction arrives.

Hierarchical address caching allows a higher cache hit ratio at the expense of address decoding each packet destination address. The improvement in cache hit ratio for 10 element LRU caches ranges from 3% to 8% and the improvement in the number-of-packets-between-cache-misses ratio ranges from 2.9 to 4.3 times better for the maximum size caches for gateways 4 and 6. Simple preprocessing before a cache for flat addresses provides a hit ratio not as high as that for hierarchical addressing, but higher than the hit ratio for flat addressing at little cost.

A simple, conservative, cost analysis shows that current gateway routing-table lookup time can be reduced to 23% of its current time. This is a conservative estimate and a good cache design or a hardware cache could further reduce the average lookup time.

6. Acknowledgments

I would like to thank Dave Clark for suggesting the idea of a routing-table cache in a gateway. My thanks also to Thu Nguyen, Tim Shepard and an anonymous reviewer for their comments.

References

[1] L. A. Belady, R. A. Nelson, and G. S. Shedler.
An Anomaly in Space-Time Characteristics of Certain Programs Running in a Paging Machine.
Communications of the ACM 12(6):349-353, June, 1969.

[2] David C. Feldmeier.
An Empirical Analysis of a Token Ring Network.
Technical Memo TM-254, MIT Laboratory for Computer Science, January, 1984.

[3] David C. Feldmeier.
Traffic Measurements on a Token Ring Network.
In *Proceedings of the 1986 Computer Networking Symposium*, pages 236-243. (Washington, DC, November, 1986).

[4] David C. Feldmeier.
Improving Gateway Performance with a Routing-Table Cache.
In *Proceedings of IEEE Infocom '88.* (New Orleans, March, 1988).
To be published.

[5] R. Jain and S. Routhier.
Packet Trains - Measurements and a New Model for Computer Network Traffic.
IEEE Journal on Selected Areas in Communications SAC-4(6):986-995, September, 1986.

[6] Alan Jay Smith.
Cache Memories.
ACM Computing Surveys 14(3):473-530, September, 1982.

OFFICIAL DISTRIBUTION LIST

Director — 2 copies
Information Processing Techniques Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Office of Naval Research — 2 copies
800 North Quincy Street
Arlington, VA 22217
Attn: Dr. R. Crafton, Code 433

Director, Code 2627 — 6 copies
Naval Research Laboratory
Washington, DC 20375

Defense Technical Information Center — 12 copies
Cameron Station
Alexandria, VA 22314

National Science Foundation — 2 copies
Office of Computing Activities
1800 G. Street, N.W.
Washington, DC 20550
Attn: Program Director

Dr. E.B. Royce, Code 38 — 1 copy
Head, Research Department
Naval Weapons Center
China Lake, CA 93555

END
DATED
FILM
8-88
DTIC